Parallels® Panel

Handbuch für Entwickler

Bereitstellungshandbuch

Parallels Small Business Panel 10.2

Parallels™

Inhalt

Einleitung 4

Typographische Konventionen ... 4
Feedback ... 5

Über dieses Handbuch 6

Anmerkungen zum Bereitstellungsworkflow 8

Verteilung des Panels 9

Einzelplatz-Verteilung ... 10
Release-ID abfragen ... 11
Liste mit den Produktkomponenten abfragen ... 12
Installieren des Panels ... 14
Installer - CLI-Befehle (Gekürzt) ... 15
Installationsskript (Linux/Unix) ... 18
Verteilung in einer Parallels Containers-Umgebung ... 20
Parallels Containers für Unix ... 21
Parallels Containers für Windows ... 25
Einrichten eines Mirrors mit dem Dienstprogramm Rsync ... 27
Mirrors von Linux/Unix-Distributionen erstellen ... 28
Mirrors von Windows-Distributionen erstellen ... 31

Überblick zu Setupvorgängen 33

Initialisierung des Panels ... 34
Installation des Lizenzschlüssels ... 35
Branding des Panels ... 36
Anpassen von Service-Links ... 36
DNS-Konfiguration ... 40
Ändern des SOA-Record-Templates ... 41
Einrichten von Templates für Ressourceneinträge (Resource Records) ... 41
Installation von SSL-Zertifikaten ... 42
Erstellen von Domains und Subdomains ... 43
Installieren von Anwendungen ... 45
Importieren eines Anwendungspakets ... 46
Installieren einer Anwendung ... 46
Verfügbare APS-Kataloge definieren ... 47
Konfigurationsdatei von APS-Katalogen ... 49

Einrichten über Remote-API 50

Remote-API verwenden ... 51
Über Remote-API ... 52
API RPC-Pakete ... 53
Erstellen der Client-Software ... 57
Einrichten des Panels ... 89

Initialisierung des Panels 89
Installation des Lizenzschlüssels 94
DNS-Konfiguration 100
Installation von SSL-Zertifikaten 128
Erstellen von Domains 135
Erstellen von Subdomains 141
Verfügbare APS-Kataloge definieren 146
Installieren von Anwendungen 150

Einrichten über CLI 207

Über die Panel-Befehlszeilenschnittstelle (CLI) 208
Einrichten des Panels 209
Initialisierung des Panels 210
Installation des Lizenzschlüssels 211
Branding des Panels 211
DNS-Konfiguration 213
Installation von SSL-Zertifikaten 220
Erstellen von Domains 222
Erstellen von Subdomains 223
Verfügbare APS-Kataloge definieren 224
Installieren von Anwendungen 225

Einleitung

In diesem Abschnitt:

Typographische Konventionen 4
Feedback 5

Typographische Konventionen

Vor der Verwendung dieses Handbuchs sollten Sie die darin verwendeten Konventionen kennen.

Mit den folgenden Formatierungen werden spezielle Informationen im Text identifiziert.

Formatierungskonvention	Informationstyp	Beispiel
Fett	Elemente, die Sie auswählen müssen, z. B. Menüoptionen, Befehlsbuttons oder Elemente einer Liste.	Gehen Sie zur Registerkarte **System**.
	Titel von Kapiteln, Abschnitten und Unterabschnitten.	Lesen Sie das Kapitel **Administrationsgrundlagen**.
Kursiv	Wird verwendet, um die Wichtigkeit einer Aussage zu betonen, einen Begriff vorzustellen oder einen Befehlszeilen-Platzhalter anzugeben, der durch einen echten Namen oder Wert zu ersetzen ist.	Das System unterstützt die so genannte Suche mit *Platzhalterzeichen*.
`Monospace`	Die Namen von Befehlen, Dateien und Verzeichnissen.	Die Lizenzdatei befindet sich im Verzeichnis `http://docs/common/licenses`.

`Vorformatiert`	Bildschirmausgabe in Befehlszeilensitzungen; Quellcode in XML, C++ oder anderen Programmiersprachen.	`# ls –al /files` `Gesamt 14470`
`Vorformatiert Fett`	Was Sie eingeben in Kontrast zur Bildschirmausgabe.	**`# cd /root/rpms/php`**
GROSSBUCHSTABEN	Namen von Tasten auf der Tastatur.	UMSCHALTTASTE, STRG, ALT
TASTE+TASTE	Tastenkombinationen, bei denen der Benutzer eine Taste drücken und gedrückt halten und dann eine weitere Taste drücken muss.	STRG+P, ALT+F4

Feedback

Wenn Sie einen Fehler in diesem Handbuch gefunden haben oder Verbesserungsvorschläge machen möchten, können Sie uns Ihr Feedback über das Onlineformular unter http://www.parallels.com/en/support/usersdoc/ zusenden. Bitte nehmen Sie in Ihren Bericht den Titel der Anleitung, des Kapitels und des Abschnitts sowie den Textabsatz auf, in dem Sie einen Fehler gefunden haben.

KAPITEL 1

Über dieses Handbuch

Zweck

In diesem Handbuch finden Sie wichtige Informationen zur Bereitstellung von Parallels Small Business Panel (nachfolgend bezeichnet als *das Panel*), das HSPs (Hosting-Service-Provider) ermöglichen soll, komplette Hosting-Lösungen für ihre Kunden bereitzustellen.

Dieses Dokument ist für HSP-Entwickler gedacht, die an der Umsetzung der Panel-Bereitstellung arbeiten, z.B. die Automatisierung der Installation/der Verteilung, die Konfiguration und die Anpassung an spezifische Kundenanforderungen.

Dieses Dokument enthält empfohlene Vorgehensweisen, die unserer Ansicht nach die besten Vorgehensweisen für die Installation und die Konfiguration des Panels im Rahmen der Produktbereitstellung sind. Diese Vorgehensweisen wurden von uns ausführlich getestet und haben sich in unseren Tests als Best Practices bewährt.Dieses Dokument enthält keinesfalls eine vollständige API-Funktionsbeschreibung: Funktionen und Optionen, die nicht wesentlich für die Bereitstellung sind, werden nicht beschrieben.

Anwendungsbereich

Heutzutage empfehlen Service-Provider weniger “Unmanaged Infrastructure Hosting” (nicht betreutes Hosting von IT-Infrastrukturen) und tendieren mehr zu Angeboten mit mehrwertigen “Managed Services” (betreute Dienste bzw. Dienstleistungen). Die Hosting-Pakete, die an Kunden verkauft werden, beinhalten nicht nur die physische oder virtuelle Serverumgebung mit optionalen Control Panel-Installationen, sondern auch Domain-Namen und SSL-Zertifikate. Da SaaS-Angebote unter Verbrauchern, insbesondere bei kleinen Unternehmen, immer beliebter werden, enthalten einige Hosting-Pakete auch schon vordefinierte Anwendungspakete, die auf Kunden-Domains installiert werden. Viele Angebote beinhalten zudem gehostete Services, wie z.B. E-Mail, Kollaborationstools und Content-Management.

Für Panel-Installationen bedeutet das, dass nach der Installation der Provider in der Lage ist, das Panel einzurichten. Das Setup des Panels beinhaltet folgende Schritte:

- Einrichten des DNS-Dienstes
- Domain-Konfiguration
- Bereitstellung von SSL-Zertifikaten
- Installation von APS-Anwendungen

Um Providern die Einführung von weiteren Diensten zu erleichtern, wird in dem vorliegenden Dokument erklärt, wie die Verteilung und Konfiguration des Panels in die Bereitstellung von Anwendungen integriert werden kann.

Überblick

Das Kapitel "Anmerkungen zum Bereitstellungsworkflow" bietet einen Einblick in die wichtigsten Phasen des Bereitstellungsworkflows.

Das Kapitel "Verteilung des Panels" konzentriert sich auf die Verteilung des Panels. Das Kapitel deckt sowohl die Installation in Parallels Containers-Umgebungen als auch in Einzelplatz-Installationen ab. Zudem beinhaltet dieses Kapitel einen ergänzenden Abschnitt, in dem näher auf die Erstellung von benutzerdefinierten Mirrors des Update-Repositorys von Parallels Small Business Panel eingegangen wird.

In den folgenden drei Kapiteln geht es um die Setup-Phase in der Panel-Bereitstellung:

- **Das Kapitel "Überblick zu Setupvorgängen"** erklärt die Bedeutung und den Umfang von Setup-Vorgängen.
- **Das Kapitel "Einrichten über Remote-API"** konzentriert sich auf das Einrichten des Panels mittels Remote-XML-API. Das Kapitel erklärt die RAPI (Remote-API): bietet einen Überblick zu dem API-RPC-Protokoll und nennt beispielhaft Tools, die verwendet werden können, um mit dem Panel-Server über das Protokoll zu interagieren. Anschließend wird im Detail beschrieben, welche Abfragenachrichten ein Drittanbieter-Tool versenden muss, um Panel-Vorgänge durchzuführen und welche Nachrichten nach jedem Vorgang empfangen werden.
- **Das Kapitel "Einrichten über CLI"** konzentriert sich auf das Einrichten des Panels über die Befehlszeilenschnittstelle des Panels (Abk. CLI, Command Line Interface). Das Kapitel bietet zuerst einen Überblick über die Grundlagen der CLI und beschreibt anschließend ausführlich, welche Befehle ausgeführt werden müssen, um Panel-Setupvorgänge durchzuführen.

KAPITEL 2

Anmerkungen zum Bereitstellungsworkflow

Bei der Bereitstellung des Parallels Small Business Panel für Windows sollten Sie zwischen der Produktverteilung und dem Setup fünf Minuten Zeit einplanen. Diese Verzögerung vor dem Setup ist notwendig, da es einige Zeit dauern kann, bis Post-Installationsskripte und Drittanbieterkomponenten eingerichtet wurden. Bereitstellungsvorgänge, die in diesem Zeitraum durchgeführt werden, könnten ansonsten fehlschlagen.

KAPITEL 3

Verteilung des Panels

In diesem Kapitel wird die Verteilungsphase in dem Bereitstellungsworkflow des Parallels Small Business Panel beschrieben.

Dieses Kapitel deckt sowohl die Installation in Parallels Containers-Umgebungen als auch Einzelplatz-Installationen ab. Darüber hinaus wird in einem zusätzlichen Abschnitt erklärt, wie Mirrors vom Update-Repositorys des Panels erstellt werden.

In diesem Kapitel:

Einzelplatz-Verteilung..... 10
Verteilung in einer Parallels Containers-Umgebung 20
Einrichten eines Mirrors mit dem Dienstprogramm Rsync 27

Einzelplatz-Verteilung

In diesem Kapitel werden die Automatisierungsmöglichkeiten bei der Verteilung des Panels in einer Umgebung ohne Parallels Containers aufgeführt (sei es Linux/Unix- oder Windows-basiert). Dieser Abschnitt des Handbuchs erklärt die Grundlagen der Panel-Installation und beinhaltet Anleitungen über und die Installation des Produkts über die Befehlszeilenschnittstelle (CLI).

Details zu den Installationsvoraussetzungen finden Sie in dem *Installationshandbuch* auf der Website mit der Dokumentation von Parallels Small Business Panel (http://www.parallels.com/products/small-business-panel/documentation/).

Die Einzelplatz-Verteilung des Panels wird über das Dienstprogramm “Parallels-Installer” durchgeführt. Das Dienstprogramm stellt eine Verbindung zu dem Update-Server des Parallels Small Business Panel her, auf dem die Panel-Pakete gespeichert werden, lädt die benötigten Pakete herunter und installiert diese.

Der Parallels-Installer kann entweder im interaktiven Modus (über die grafische Weboberfläche oder die Befehlszeile) oder automatisch verwendet werden, wenn er über die Befehlszeile mit bestimmten Optionen ausgeführt wird. Im letzteren Fall kann die Panel-Verteilung mithilfe von Shell-Skripten automatsiert werden.

Es existieren verschiedene Distributionen des Parallels-Installers für unterschiedliche Betriebssysteme und Plattformen. Bitte vergewissern Sie sich aus diesem Grund, dass Sie das Parallels-Installationsprogramm verwenden, das für Ihre Host-Betriebssysteme entwickelt wurde. Im Normalfall werden die Binärdateien des Parallels-Installers in dem folgenden Format verteilt:
`parallels_installer_v<Installer-Version>_os_<Version des Betriebssystems>_<Plattform>`.

Hinweis: Wenn die Befehlszeile des Parallels-Installers beschrieben wird, gehen wir davon aus, dass das Dienstprogramm “Parallels-Installer” folgendermaßen heißt: `parallels_installer`.

Die offiziellen Parallels Small Business Panel-Repositorys sind autoinstall.plesk.com (Linux/Unix-Pakete) und autoinstall-win.parallels.com (Windows-Pakete). Sie sollten benutzerdefinierte Mirrors (auf Seite 27) der Panel-Repositorys innerhalb Ihrer Hosting-Umgebung einrichten, um Zeit und Traffic zu sparen, und um die Installation zu sichern.

In diesem Abschnitt:

Release-ID abfragen 11
Liste mit den Produktkomponenten abfragen 12
Installieren des Panels 14
Installer - CLI-Befehle (Gekürzt) 15
Installationsskript (Linux/Unix) 18

Release-ID abfragen

Die Release-ID ist eine eindeutige ID der Panel-Version, zum Beispiel Parallels Small Business Panel 10.2.

Führen Sie den folgenden Befehl aus, um eine Liste mit allen Release-IDs abzufragen:

```
parallels_installer—show-all-releases
```

auf einem Linux/Unix-Server

```
parallels_installer.exe—show-all-releases
```

auf einem Windows-Server

Es öffnet sich eine Liste mit den Produkt-Versionen. Beispiel:

- PPSMBE_10_2_0 (Parallels Small Business Panel 10.2.0)

Liste mit den Produktkomponenten abfragen

Um eine Liste mit den Komponenten dieser Release zu erhalten, führen Sie den Befehl in dem folgenden Format aus:

parallels_installer—select-release-id <*ID*> --show-components

Das bedeutet für Parallels Small Business Panel 10.2:

```
parallels_installer—select-release-id PPSMBE_10_2_0 --show-components
```

auf einem Linux/Unix-Server

```
parallels_installer.exe—select-release-id PPSMBE_10_2_0_WIN—show-components
```

Eine Liste mit den Komponenten wird im linken Teil angezeigt und eine Angabe darüber, ob diese Komponenten installiert (`[install]`) oder aktualisiert (`[upgrade]`), mit einer Kurzbeschreibung auf der rechten Seite, zum Beispiel:

- base [install] - Parallels Small Business Panel-Basispakete
- autoinstaller [install] - Installationsprogramm für Parallels-Produkte
- postfix [install] - Postfix-Mailserver
- php5 [install] - PHP5-Unterstützung
- postgresql [install] - PostgreSQL-Unterstützung
- api [install] - Parallels Small Business Panel-API
- siteeditor [install] - Site Editor
- setemplates [install] - Site Editor-Templates
- mod_python [install] - Apache mod_python-Modul
- ruby [install] - Ruby on Rails-Unterstützung
- firewall [install] - Firewall-Add-on
- vpn [install] - VPN-Add-on
- fileserver [install] - Fileserver-Add-on
- watchdog [install] - Add-on für die Systemüberwachung (Watchdog)
- drweb [install] - Parallels Premium Antivirus
- spamassassin [install] - SpamAssassin-Unterstützung
- backup [install] - Backup-Manager
- backup-vz [install] - Backup-Manager für die PVC-Neuinstallation
- de-DE-locale [install] - Sprachpaket für Deutsch
- ja-JP-locale [install] - Sprachpaket für Japanisch
- es-ES-locale [install] - Sprachpaket für Spanisch
- ru-RU-locale [install] - Sprachpaket für Russisch
- fr-FR-locale [install] - Sprachpaket für Französisch
- it-IT-locale [install] - Sprachpaket für Italienisch
- zh-CN-locale [install] - Sprachpaket für Chinesisch (China)
- zh-TW-locale [install] - Sprachpaket für Chinesisch (Taiwan)
- nl-NL-locale [install] - Sprachpaket für Niederländisch

- atmail [install] - Atmail Webmail-Unterstützung

Installieren des Panels

Um Parallels Small Business Panel zu installieren, führen Sie den folgenden Befehl im folgenden Format aus:

parallels_installer <Quell-Optionen für Pakete> `--select-release-id <ID>` *<Installationsoptionen für Komponenten> [andere Optionen]*

wobei

- "Quell-Optionen für Pakete" geben Sie den Ort an, wo das Installationsprogramm die Panel-Pakete für die Installation finden kann

```
<Quell-Optionen für Pakete> =
•  source <URL>
falls Sie einen Mirror des Panel-Update-Servers verwenden
_ODER
•  source <Pfad>
falls Sie ein lokales Dateisystem verwenden
_ODER
Keine
falls Sie die Installation über den Panel-Update-Server durchführen
```

- "Installationsoptionen (auf Seite 12)für Komponenten" definiert welche Panel-Komponenten installiert werden sollen

```
<Installationsoptionen für Komponenten> =
•  install-everything
_ODER
•  install-component Komponente1 [--install-component Komponente2
[... [--install-component KomponenteN]]]
```

- andere Optionen beinhalten die Konfigurationen von Proxy-Einstellungen, das Installationsprotokoll und so weiter. Weitere Details finden Sie im Abschnitt **Installer - CLI-Befehle**. (auf Seite 15)

Beispiele: Linux/Unix

1. Mithilfe des folgenden Befehls installieren Sie Parallels Small Business Panel 10.2 (Release-ID ist `PPSMBE_10_2_0`) über den Mirror des Servers, der über HTTP auf dem Host ppsmbe-mirror.example.com zur Verfügung steht. Die Installationsdateien werden vorübergehend unter /tmp/ppsmbe gespeichert und der Statusbericht der Installation wird an die E-Mail-Adresse admin@example.com versendet. Die installierten Komponenten sind die Basispakete des Panels, der PostgreSQL-Server und der Spamfilter von SpamAssassin.

```
./parallels_installer—source http://ppsmbe-mirror.example.com/ --target
/tmp/ppsmbe—select-release-id PPSMBE_10_2_0 --install-component
base—install-component postgresql—install-component
spamassassin—notify-email admin@example.com
```

2. Mit den folgenden Befehlen wird die Installation (alle verfügbaren Produktkomponenten werden installiert) von Parallels Small Business Panel 10.2 (Release-ID `PPSMBE_10_2_0`) über den Parallels-Update-Server abgeschlossen. Eine Ausgabedatei des Installers wird im XML-Format erstellt.

```
./parallels_installer—select-release-id PPSMBE_10_2_0
--install-everything—enable-xml-output
```

Beispiele: Windows

1.Mithilfe des folgenden Befehls installieren Sie Parallels Small Business Panel 10.2 (Release-ID ist PPSMBE_10_2_0_WIN) über den Mirror des Servers, der über HTTP auf dem Host ppsmbe-mirror.example.com zur Verfügung steht. Die Installationsdateien werden vorübergehend in dem Ordner `%SystemDrive%\Parallels\` gespeichert und der Statusbericht der Installation wird an die E-Mail-Adresse admin@example.com versendet. Folgende Komponenten werden installiert: Panel-Basispakete, Bind-Nameserver, MailEnable-Mailserver, PHP Scripting Engine, Site Editor und Horde Webmail.

```
parallels_installer.exe—source http://ppsmbe-mirror.example.com/ --target
%SystemDrive%\Parallels—select-release-id
PPSMBE_10_2_0_WIN—install-component base—install-component
dns—install-component mailenable—install-component php5 --install-component
siteeditor—install-component webmail—notify-email admin@example.com
```

2.Mit den folgenden Befehlen wird die Installation (alle verfügbaren Produktkomponenten werden installiert) von Parallels Small Business Panel 10.2 (Versions-ID PPSMBE_10_2_0_WIN) über den Parallels-Update-Server abgeschlossen. Eine Ausgabedatei des Installers wird im XML-Format erstellt.

```
parallels_installer.exe—select-release-id
PPSMBE_10_2_0_WIN—install-everything—enable-xml-output
```

Installer - CLI-Befehle (Gekürzt)

In diesem Abschnitt werden die CLI-Befehle des Parallels-Installers aufgeführt, die nur für die Installation des Panels über die Befehlszeilenschnittstelle (CLI) relevant sind.

Sie erhalten weitere Informationen zu Optionen, die wichtig für ein Upgrade eines bereits installierten Panels oder des Host-Betriebssystems sind, indem Sie den folgenden Befehl ausführen:

parallels_installer—help

Option	Beschreibung und mögliche Argumente
`--source` *<Pfad>*|*<URL>*	Wenn Sie die Panel-Pakete von einem lokalen Dateisystem abfragen, geben Sie die Option—`source` ein, um auf die Datei `.inf3` zu verweisen, die die Informationen zu dem zu installierenden Panel-Build enthält. Wenn Sie die Panel-Pakete von einem Netzwerk-Server abfragen, geben Sie die Option `--source` *<URL>* ein, um auf das Verzeichnis zu verweisen, wo sich die Kopie (Mirror) des Panel-Update-Servers befindet.

Option	Beschreibung und mögliche Argumente

`--target <Pfad>`	Standardmäßig speichert der Installer die abgefragten Dateien in dem Verzeichnis `/<aktueller Benutzername>/psa`. Beispiel: Wenn der Installer von einem Benutzer über den Root-Account ausgeführt wurde, wird z.B. das Verzeichnis `/root/psa` erstellt und verwendet. Falls Sie ein benutzerdefiniertes Verzeichnis zum Speichern der abgefragten Dateien festlegen wollen, verwenden Sie die Option—`target`.
	Beispiel:
	`--target /opt/storage/psa`
	`_ODER`
	`--target D:\temp`

`--proxy-host <Netzwerkadresse>`	Wenn Sie einen Proxy-Server oder eine Firewall benutzen, sollten Sie diese Option verwenden, um den Domain-Namen eines Proxy-Servers oder die IP-Adresse festzulegen. Beispiel: `--proxy-host proxy.example.com`
`--proxy-port <Port-Nummer>`	Der Standard-Port für die Verbindung zum Proxy-Server ist: 3128. Falls Ihr Proxy-Server eine andere Port-Nummer verwendet, sollten Sie diese angeben, indem Sie diese Option verwenden. Beispiel: `--proxy-port 5741`

Optionen für die Proxy-Athentifizierung:
`--proxy-user <Benutzername>`
`--proxy-password <Passwort>`
	Falls für Ihren Proxy-Server eine Authentifizierung erforderlich ist, sollten Sie diese Optionen verwenden, um den Installer beim Proxy-Server zu authentifizieren.
	Beispiel:
	`--proxy-user smith—proxy-password f1sZ9AnQ4EnO52`

`--show-releases`	Geben Sie diese Option an, um eine Liste mit den verfügbaren Versionen für das Betriebssystem abzufragen, wo der Installer ausgeführt wird. Sie erhalten eine Liste mit Release-Identifiers und Versionsbeschreibungen. Die Release-IDs sind die Informationen, mit denen Sie sich beschäftigen sollten.
`--show-all-releases`	Diese Option zeigt alle Versionen an, die über den Panel-Updates-Server zur Verfügung stehen.
`--select-release-id <ID>`	Verwenden Sie diese Option, um die Version anzugeben, die installiert werden soll oder deren Eigenschaften angezeigt werden sollen.
`--select-release-latest`	Verwenden Sie diese Option, um die neueste - für Ihr Betriebssystem verfügbare - Version auszuwählen.
`--show-components`	Geben Sie diese Option ein, um Informationen darüber zu erhalten, welche Komponenten für die ausgewählte Version verfügbar sind. Es werden die Komponentenbeschreibungen und Namen angezeigt. Um weitere zu installierende Komponenten auszuwählen, müssen Sie die Namen der Komponenten angeben.
`--install-component <Name-der-Komponente>`	Verwenden Sie diese Option, um eine zu installierende Komponente anzugeben. Falls mehr als zwei Komponenten auf einmal installiert werden sollen, wiederholen Sie diese Option für jede Komponente. Beispiel: `--install-component base—install-component postgresql—install-component spamassassin`

Option	Beschreibung und mögliche Argumente
`--install-everything`	Verwenden Sie diese Option, um alle Komponenten der ausgewählten Version zu installieren.
`--show-os-list`	Verwenden Sie diese Option, um herauszufinden, welche Betriebssysteme von der ausgewählten Panel-Version unterstützt werden.
`--no-space-check`	Installiert Pakete, auch wenn wenig Festplattenplatz zur Verfügung steht.
`--no-daemon`	Führt den Installationsprozess im Vordergrund aus.
`--notify-email` *<E-Mail>*	Verwenden Sie diese Option, damit der Installer Ihnen den Statusbericht per E-Mail zusendet. Ein Bericht über die erfolgreiche Fertigstellung beinhaltet ein detailliertes Protokoll und eine Liste mit den installierten/aktualisierten Paketen.
`--enable-xml-output`	Diese Option wurde für die Kommunikation mit dem Panel konzipiert und kann für die Interaktion mit anderen Anwendungen verwendet werden. Wenn Sie diese Option angeben, wird die gesamte Ausgabe des Installers in XML eingebunden. Bitte beachten Sie, dass dieser Code es nicht zulässt, dass der Installer Fehler über den Exitcode mitteilt. Der Exitcode ist immer Null und alle Fehler werden innerhalb der XML-Ausgabe mitgeteilt.
`--query-status`	Da die RPM-Datenbank nicht mehrere Zugriffe zulässt, sperrt der Installer diese Funktionalität. Führen Sie diese Option aus, um herauszufinden, ob der Installer aktiv ist. Diese Option überprüft, ob eine Sperre vorhanden ist und erstellt entweder einen Exitcode (`0` - der Installer ist inaktiv, `1` - der Installer wird ausgeführt) oder eine XML-formatierte Ausgabe.
`--truncate-log`	Löscht die Protokolldatei beim Start des Installationsprogramms für Parallels-Produkte.
`--separate-log`	Verwendet eine neue Protokolldatei, wenn das Installationsprogramm für Parallels-Produkte neu gestartet wird.
`--debug`	Aktiviert eine detaillierte Ausgabe in der Protokolldatei.
`--version`	Zeigt die Version des Installationsprogramms für Parallels-Produkte an.

Installationsskript (Linux/Unix)

In diesem Abschnitt erhalten Sie Informationen zur Implementierung und Ausführung eines Installationsskripts. Zudem werden zwei Beispiele für Installationsskripte angegeben.

Hinweise zur Implementierung und Ausführung

- Damit Sie nicht jedes Mal nach einem Passwort gefragt werden, wenn Sie sich über SSH mit den Servern verbinden, auf denen das Panel installiert werden soll, sollten Sie Ihren öffentlichen Schlüssel zu der Liste mit den authorisierten Schlüsseln des Benutzers hinzufügen, für den das Skript auf dem Server ausgeführt werden soll (weitere Informationen zu der Vorgehensweise finden Sie auch in der SSH-Dokumentation).
- Falls Sie einen lokalen Mirror des Panel-Update-Servers benutzen (wie im Abschnitt **Einrichten von Mirrors** beschrieben (auf Seite 27)), verwenden Sie die folgende Option:
  - `source <Mirror-URL>`
- Standardmäßig werden alle heruntergeladenen Pakete in dem Verzeichnis `/root/psa` gespeichert. Um ein anderes Verzeichnis auszuwählen, verwenden Sie bitte die folgende Option:
  - `target <Name-des-Verzeichnisses>`

Beispiel-Skripte

1.Dieses Beispiel-Skript kann dann angewendet werden, wenn der Parallels-Installer zuvor auf den Zielserver hochgeladen wurde und das Ausführungs-Bit in den Zugriffsrechten definiert wurde.

#!/bin/sh

SERVERS_LIST=”node1.example.com node2.example.com”

for current_server in $SERVERS_LIST; do

scp parallels_installer root@$current_server:

ssh -f root@$current_server “parallels_installer—source http://updates.example.com/ --target /tmp/ppsmbe—select-release-id PPSMBE_10_2_0 --install-component base—install-component postgresql—install-component asp—notify-email admin@example.com”

Done

2. Dieses Beispiel-Skript kann angewendet werden, wenn der Binärcode des Parallels-Installers direkt über eine Netzwerkadresse abgerufen wird (die Beispiel-URL http://example.com/type_parallels_installer_name_here sollte durch einen gültigen Download-Link zum Installer ersetzt werden).

#!/bin/sh

SERVERS_LIST=”node1.example.com node2.example.com”

for current_server in $SERVERS_LIST; do

ssh -f root@$current_server ‘wget http://example.com/type_parallels_installer_name_here -o parallels_installer;chmod 755 ./parallels_installer;./parallels_installer—source http://updates.example.com/ --target /tmp/ppsmbe—select-release-id PPSMBE_10_2_0 --install-component base—install-component postgresql—install-component spamassassin—notify-email admin@example.com’

Done

Verteilung in einer Parallels Containers-Umgebung

Dieser Abschnitt beschäftigt sich mit den grundlegenden Schritten, die durchgeführt werden müssen, wenn das Panel mithilfe von Parallels Containers verteilt wird. Der Schwerpunkt liegt dabei auf der Erstellung eines Containers mit Parallels Small Business Panel. Weitere Informationen zu der Parallels Containers-CLI oder der der Parallels Containers-API (XML oder SOAP) finden Sie in der Parallels Containers- Entwicklerdokumentation (http://www.parallels.com/ptn/documentation/virtuozzo/).

Für die Verteilung des Panels in einer Parallels Containers-Umgebung sind die folgenden zwei Schritte notwendig:

1. Installation des Panel-Anwendungs-Templates auf einem Hardware-Node.
2. Erstellen eines Containers und Verteilung der Anwendung über das Template.

Der erste Schritt wird auf jedem Hardware-Node nur einmal ausgeführt: Sobald ein Anwendungs-Template auf einem Hardware-Node installiert wurde, kann man dieses auf beliebig viele Container verteilen.

Sie müssen nicht jedes Mal, wenn Sie einen Container mit dem Panel verteilen müssen, einen neuen Container erstellen. Sie können stattdessen einen existierenden Container klonen, in dem das Panel bereits installiert wurde - dieser Container fungiert somit als Template Container.

In diesem Abschnitt:

Parallels Containers für Unix 21
Parallels Containers für Windows 25

Parallels Containers für Unix

Es stehen zwei EZ-Templates für Parallels Small Business Panel 10.2 zur Verfügung:

- `ppsmbe10`
  Enthält die Kernkomponenten des Panels.
- `ppsmbe10-extended`
  Enthält alle zusätzlichen Komponenten, wie z.B. SpamAssassin-Spamfilter, Ruby-on-Rails Framework, Parallels Premium Antivirus, weitere Site Editor-Templates und andere Komponenten.

Falls eine erweiterte Version des Panels installiert werden soll, installieren Sie beide Templates.

Für die Installation eines Panel-Templates auf dem Ziel-Hardware-Node können Sie eine der folgenden zwei Methoden wählen:

- Verwenden Sie das Dienstprogramm `vzup2date`. Weitere Details finden Sie in dem *Referenzhandbuch: Parallels Containers für Linux* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).
- Laden Sie sich das Template direkt von der Parallels-Website (http://www.parallels.com) herunter und installieren Sie es mit dem Dienstprogramm `vzpkg`.

Wenn das Template auf einem Hardware-Node installiert wird, werden die wichtigsten Dateien verteilt, dazu zählt die Definition des Repositorys für Anwendungspakete. Sollte die Anwendung anschließend in einem Container installiert werden, kommuniziert Parallels Containers mit dem Repository (das Standard-Repository für Parallels Small Business Panel für Linux/Unix ist der Parallels-Updates-Server unter http://autoinstall.plesk.com) und fragt dort die Anwendungspakete ab und installiert diese in dem Container. Die Pakete werden so lange im Cache des Nodes gespeichert bis dieser geleert wird. Dann werden die Pakete wieder über das Repository abgefragt, wenn die Anwendung in dem Container installiert wird. Um Traffic und Zeit zu sparen, sollten Sie einen benutzerdefinierten Mirror des Panel-Repositorys verwenden.

➢ *So installieren Sie das Panel-Template auf einem Hardware-Node von Parallels Containers unter Verwendung des Dienstprogramms vzpkg:*

1. Beziehen Sie ein Panel-Template über die Parallels-Website (http://www.parallels.com) and und laden Sie diese auf den Ziel-Hardware-Node von Parallels Containers hoch.
2. Installieren Sie das Anwendungs-Template, indem Sie den Befehl im folgenden Format ausführen:

   ```
   vzpkg install template [-q|<-d <Nummer>>] <Pfad_zum_Paket> ...
   ```

   wobei

   - `-q` die Protokollierung an das Display und die Protokolldatei deaktiviert
   - `-d|--debug <Nummer>` den Ausführlichkeitsgrad des Protokolls von 0 auf 10 setzt

Beispiel: Der folgende Befehl installiert sowohl die Kern- als auch die Zusatz-Komponenten des Panels:

```
# vzpkg install template
ppsmbe10-debian-5.0-x86-ez-4.0.0-1.prl.284629.noarch.rpm
ppsmbe10-extended-debian-5.0-x86-ez-4.0.0-1.prl.284385.noarch.rpm
```

➢ *So verwenden Sie einen benutzerdefinierten Mirror:*

1. Erstellen Sie einen Mirror wie in dem Abschnitt **Mirrors von Linux/Unix-Distributionen erstellen** beschrieben (auf Seite 28).
2. Bearbeiten Sie auf dem Hardware-Node die Datei, die die URL des Repositorys deklariert:
   a. Öffnen Sie zum Bearbeiten die Datei `/vz/template/<Betriebssystem>/<Version-des-Betriebssystems>/<Plattform>/config/app/<Name-des-Anwendungs-Templates>/default/repositories`.

      Das bedeutet `/vz/template/<Betriebssystem>/<Version-des-Betriebssystem>/<Plattform>/config/app/ppsmbe10/default/repositories` und `/vz/template/<Betriebssystem>/<Version-des-Betriebssystems>/<Plattform>/config/app/ppsmbe10-extended/default/repositories`. Bearbeiten Sie beide, falls beide Templates installiert wurden.
   b. Ersetzen Sie die URLs des Parallels-Updates-Server mit den entsprechenden URLs des Mirrors und speichern Sie die Datei.
3. Bereinigen oder rufen Sie die Metadaten der Pakete neu ab.

   Sie können hierzu beispielsweise den Befehl `vzpkg clean` verwenden. Weitere Details finden Sie in dem *Referenzhandbuch: Parallels Containers für Linux* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).

➢ *So erstellen Sie einen neuen Container und installieren das Panel in diesem:*

1. Erstellen Sie einen Container und konfigurieren Sie diesen gemäß Ihren Anforderungen (optional).

   Führen Sie die folgenden Befehle aus:

   ```
   vzctl create <CT_ID> [Optionen]
   vzctl set <CT_ID> <Name_der_Einstellung> <Wert> [--save]
   ```

   wobei

   - `<CT_ID>` eine beliebige Container-ID höher als 100 definiert, die auf dem Hardware-Node eindeutig ist
   - `--save` Switch teilt `vzctl` mit, ob Änderungen in der Konfigurationsdatei des Containers gespeichert werden sollen oder nicht

Hinweis: Weitere Informationen zur Erstellung von Containern und zu Konfigurationsoptionen finden Sie im *Referenzhandbuch: Parallels Containers für Linux* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).

Zum Beispiel können Sie mithilfe der unten aufgeführten Befehle folgende Vorgänge durchführen:

1. erstellen Sie einen Container mit der ID 444 mit der IP 192.0.2.44, basierend auf dem Standard-Betriebssystem-Template, das in der globalen Konfigurationsdatei von Parallels Containers definiert wurde
2. legen Sie Grenzwerte und Limits für folgende Parameter fest: nicht austauschbaren Kernel-Speicher, privater (oder möglicherweise privater) Speicher, Anzahl der Dateien geöffnet durch alle Container-Prozesse, Speicherplatz und die Gesamtanzahl an Disk-Inodes (Dateien, Verzeichnisse, symbolische Links) fest, die ein Container zuweisen kann
3. speichern Sie Grenzwerte/Limits in der Konfigurationsdatei des Containers
4. aktivieren Sie die Verwaltung des Containers unter Verwendung eines Webbrowsers (deaktivieren Sie die Offline-Verwaltung mithilfe der Option "`–offline_management no`")

Hinweis: Es ist äußerst wichtig, dass Sie diese Option exakt wie im obigen Beispiel verwenden. Andernfalls können Sie nicht über den Webbrowser auf das Panel zugreifen.

```
# vzctl create 444 --ipadd 192.0.2.44
# vzctl set 444 --save–kmemsize 24299200:26429120 --privvmpages
362144:392912 --numfile 12000:12000 --diskspace 5117880:5242880
--diskinodes 350000:370000 offline_management no
```

2. Starten Sie den neu erstellten Container:

```
# vzctl start <CT_ID>
```

3. Installieren Sie das Panel-Template in dem Container:

```
vzpkg install <CT_ID> <ppsmbe_Name_des_Templates> ...
```

Beispiel: Der folgende Befehl installiert sowohl die Kern- und Zusatz-Komponenten des Panels:

```
# vzpkg install 444 ppsmbe10 ppsmbe10-extended
```

➢ *So klonen Sie einen Container:*

Führen Sie den Befehl im folgenden Format aus:

vzmlocal -C {CT List}

{CT List} = <source_CT_ID>:<dst_CT_ID>[:[<dstCT_private>][:<dstCT_root>]] [...]

Sie müssen die Quell-Container-ID (*<source_CT_ID>*) und die Ziel-Container-ID *(<dest_CT_ID>*) angeben. Optional können Sie den "Private Area Path" des Ziel-Containers *(<dstCT_private>*) und den root-Pfad (*<dstCT_root>*) angeben; dies ermöglicht Ihnen, die standardmäßig festgelegten Pfade - `/vz/private/`*<dst_CT_ID>* und */vz/root/<dst_CT_ID>* zu überschreiben. Weitere Informationen zu Befehlsoptionen finden Sie im *Referenzhandbuch: Parallels Containers für Linux* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).

Beispiel: So klonen Sie einen Container, der mithilfe des Panels während des vorigen Vorgangs erstellt wurde:

```
# vzmlocal -C 444:445
```

Parallels Containers für Windows

➢ *So installieren Sie das Panel-Template auf einem Hardware-Node von Parallels Containers:*

1. Beziehen Sie ein Panel-Template über die Parallels-Website (http://www.parallels.com) und laden Sie diese auf den Ziel-Hardware-Node von Parallels Containers hoch.
2. Installieren Sie das Anwendungs-Template des Panels. Falls Sie die Installation auf Microsoft Windows Server 2003 durchführen, installieren Sie außerdem Microsoft .NET Framework v.2.0 oder später, indem Sie den Befehl im folgenden Format ausführen:

   ```
   vzpkgdeploy [-q|-v] -i <Template-der-Datei>
   ```

   wobei

   - `-q` die Protokollierung an das Display und die Protokolldatei deaktiviert
   - `-v` das Protokoll-Level auf den höchstmöglichen Wert für diese `vzpkgdeploy`-Sitzung festlegt

   Beispiel:

   ```
   >vzpkgdeploy -i ppsmbe_10.2.0_20091126.22.efd
   >vzpkgdeploy -i dotnet2.0_frmwk-2.0.50727.42.efd
   ```

Hinweis: Die Installation eines weiteren Templates des .NET Frameworks ist nur auf einem Windows Server 2003 (PVC 4.0, VZ 3.5.1) notwendig. Dieser Vorgang ist nicht beim Windows Server 2008 (PVC 4.5) erforderlich.

➢ *So erstellen Sie einen neuen Container und installieren das Panel in diesem:*

1. Erstellen Sie einen Container und konfigurieren Sie diesen gemäß Ihren Anforderungen (optional).

   Führen Sie die folgenden Befehle aus:

   ```
   vzctl create <CT_ID> --pkgset name [Optionen]
   vzctl set <CT_ID> <Name_der_Einstellung> <Wert> [--save]
   ```

   wobei

   - *<CT_ID>* eine beliebige Container-ID höher als 100 definiert, die eindeutig auf dem Hardware-Node ist
   - `--pkgset` *name* das Betriebssystem-Template bezeichnet, das für die Erstellung des Containers verwendet werden soll
   - der Switch—`save` `vzctl` mitteilt, ob Änderungen in der Konfigurationsdatei des Containers gespeichert werden sollen oder nicht

Hinweis: Weitere Informationen zur Erstellung von Containern und zu Konfigurationsoptionen finden Sie im *Referenzhandbuch: Parallels Containers für Windows* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).

Zum Beispiel können Sie mithilfe der unten aufgeführten Befehle folgende Vorgänge durchführen:

- erstellen Sie einen Container mit der ID 444 mit der IP 192.0.2.44, basierend auf dem Betriebssystem-Template w2k3
- legen Sie eine Speicherplatzbeschränkung von 4+ Gigabytes fest und setzen Sie das Limit für den privaten Speicher auf 1 Gigabyte und das Administrator-Passwort auf "P4$$w0rd"
- aktivieren Sie die Verwaltung des Containers unter Verwendung eines Webbrowsers (deaktivieren Sie die Offline-Verwaltung mithilfe der Option "`—offline_management no`")

Hinweis: Es ist äußerst wichtig, dass Sie diese Option exakt wie im obigen Beispiel verwenden. Andernfalls können Sie nicht über den Webbrowser auf das Panel zugreifen.

```
>vzctl create 444 --pkgset w2k3 --ipadd 192.0.2.44
>vzctl set 444 --save—diskspace 4500000 --vprvmem 1024 --userpasswd
Administrator:P4$$w0rd—offline_management no
```

2. Starten Sie den neu erstellten Container:

```
>vzctl start <CT_ID>
```

3. Installieren Sie Microsoft .NET Framework Version 2.0 oder später in Ihrem Container:

```
vzpkgadd <CT_ID> <dotnet_template_name>
```

Beispiel:

```
>vzpkgadd 444 dotnet2.0_frmwk-2.0.50727.42/20070613
```

4. Installieren Sie das Template für Parallels Small Business Panel in dem Container:

```
vzpkgadd <CT_ID> <ppsmbe_template_name>
```

Beispiel:

```
>vzpkgadd 444 ppsmbe_10.2.0_20091126.22
```

➢ *So klonen Sie einen Container:*

Führen Sie den Befehl im folgenden Format aus:

vzmlocal -C <CT_List>

<CT_List> = <source_CTID>:<dest_CTID>[:<dest_private>] [Optionen]

Sie müssen die Quell-Container-ID (*<source_CTID>*) und die Ziel-Container-ID *(<dest_CTID>*) angeben. *Optional können Sie den "Private Area Path" zum Ziel-Container (<dest_private>) angeben;* Sie können so den Standard-Pfad von `X:\vz\private\<CT_ID>` überschreiben. Weitere Informationen zu Befehlsoptionen finden Sie im *Referenzhandbuch: Parallels Containers für Windows* (verfügbar im Parallels Technology Network (http://www.parallels.com/ptn/documentation/virtuozzo/)).

Beispiel: So klonen Sie einen Container, der mithilfe des Panels während des vorigen Vorgangs erstellt wurde:

```
>vzmlocal -C 444:445
```

Einrichten eines Mirrors mit dem Dienstprogramm Rsync

Dieser Abschnitt beschreibt, wie Sie einen lokalen Mirror des Parallels-Updates-Servers sowohl für Linux/Unix- als auch für Windows-Distributionen des Parallels Small Business Panel auf einem Linux-Server mithilfe von `Rsync` erstellen. `Rsync` ist ein Open-Source-Dienstprogramm, welches die Erstellung von Mirrors ermöglicht, die ein gewünschtes Set an Panel-Distributionen (inklusive denen für Parallels Containers) unterstützten.

Ein Mirror von Parallels Small Business Panel für ein Linux-Repository unterstützt sowohl Einzelplatz- als auch Parallels Containers-Installationen (abhängig von der Anzahl der kopierten Pakete).

In diesem Abschnitt:

Mirrors von Linux/Unix-Distributionen erstellen 28
Mirrors von Windows-Distributionen erstellen 31

Mirrors von Linux/Unix-Distributionen erstellen

Das Repository von Parallels Small Business Panel für Linux/Unix befindet sich unter autoinstall.plesk.com und ist wie folgt strukturiert (es werden nur die Dateien und Verzeichnisse aufgelistet, die für das Erstellen von Mirrors der Panel-Pakete relevant sind):

- *<PRODUKTNAME>_<Produkt_Version>/*

  Mehrere Unterverzeichnisse werden in Übereinstimmung mit den Panel-Versionen genannt.

  In unserem Fall sind folgende Unterverzeichnisse interessant

  a PPSMBE_*<Version>*/

  z.B., PPSMBE_10.2.0

  wo sich die Panel-Pakete befinden

  b SETEMPLATES_*<Version>*/

  z.B., SETEMPLATES_1.0.0

  wo Pakete für mehr als 200 weitere Site-Design-Templates für die Site Editor-Komponente gespeichert sind. Es handelt sich eigentlich nicht um ein Produkt, sondern um eine Komponente des Panels, die sich in einem separaten Produktverzeichnis befindet, um Speicherplatz und Traffic zu sparen.

  Jedes dieser Unterverzeichnisse enthält die folgenden Dateien:

  - dist-<Typ>-<Name-des-Betriebssystems>-<Version>-<Architektur>/

    Enthält Distributionspakete des Panels oder die Design-Templates

    z.B., dist-deb-Debian-5.0-x86_64/ oder dist-deb-Debian-all-all/

  - update-*<Typ>*-*<Name-des-Betriebssystems>*-*<Version>*-*<Architektur>*

    Enthält System-Udates für das Betriebssystem des Servers; *<Typ>* bezeichnet den Pakettyp: rpm, deb, pkg

  - thirdparty-*<Typ>*-*<Name-des-Betriebssystems>*-*<Version>*-*<Architektur>*

    Enthält weitere Pakete von Drittanbietern

  - *<PRODUKTNAME>_<Parallels-Panel-Version>-<Name-des-Betriebssysteme>-<Architektur>.inf3*

    Konfigurationsdateien des Parallels-Installers

    z.B., ppsmbe-10.2.0-suse11.1-x86_64.inf3, setemplates-1.0.0-deball-all.inf3

- products.inf3
  Die Konfigurationsdatei des Parallels-Installers, welche die Panel-Produkte beschreibt
- ppsmbe.inf3, setemplates.inf3
  Konfigurationsdateien des Parallels-Installers, die unterschiedliche Versionen des Panels und weitere Komponenten beschreiben
- PPSMBE10/
  Parallels Small Business Panel-Metadaten, die für die Installation des Panels innerhalb von Parallels Containers mithilfe von EZ-Templates benötigt werden.

- `debian/`, `ubuntu/`
  Verzeichnisse, die als `apt-get`-Repositorys für die Installation von Parallels-Produkten mithilfe von EZ-Templates und dem Parallels-Installer verwendet werden

➢ *So erstellen Sie einen Mirror für das Panel:*

1. Melden Sie sich an dem Server an, auf dem der Mirror eingerichtet werden soll.
2. Erstellen Sie ein Verzeichnis, in dem die Dateien abgelegt werden sollen. Zur Veranschaulichung nennen wir dieses Verzeichnis `destination_directory/`.
3. Erstellen Sie in diesem Verzeichnis zwei Unterverzeichnisse: `debian/` und `ubuntu/`.
4. Führen Sie die folgenden Befehle aus, um einen Mirror einzurichten.

   - Mit diesem Befehl können Sie das Verzeichnis /`PPSMBE_10.2.0` aus dem Parallels-Updates-Repository in das Verzeichnis `destination_directory` herunterladen.

```
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/PPSMBE_10.2.0
destination_directory
```

   - Mit diesem Befehl können Sie das Verzeichnis /`SETEMPLATES_1.0.0` aus dem Parallels-Updates-Repository in das Verzeichnis `destination_directory` herunterladen.

```
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/SETEMPLATES_1.0.0
destination_directory
```

   - Mit diesen Befehlen können Sie die Inhalte herunterladen, die für die Installation der Ubuntu- und Debian-Pakete von Parallels Small Business Panel mithilfe des Parallels-Installers benötigt werden.

```
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/debian/PPSMBE_10.2
.0 destination_directory/debian
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/ubuntu/PPSMBE_10.2
.0 destination_directory/ubuntu
```

   - Mit diesen Befehlen können Sie die Inhalte herunterladen, die für die Installation der Ubuntu- und Debian-Pakete für weitere Site Editor-Templates mithilfe des Parallels-Installers benötigt werden.

```
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/debian/SETEMPLATES
_1.0.0 destination_directory/debian
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/ubuntu/SETEMPLATES
_1.0.0 destination_directory/ubuntu
```

- Mit diesen Befehlen können Sie die Inhalte herunterladen, die für die Installation der Ubuntu- und Debian-Pakete des Panels mithilfe der Parallels Containers EZ-Templates benötigt werden. Falls Sie einen Mirror für eine Umgebung ohne Parallels Containers einrichten, lassen Sie diesen Schritt aus.

```
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/debian/PPSMBE_10.2
.0/ destination_directory/debian/PPSMBE10
# rsync -au—delete
rsync://rsync.autoinstall.plesk.com/autoinstall/ubuntu/PPSMBE_10.2
.0/ destination_directory/ubuntu/PPSMBE10
```

- Dieser Befehl erstellt einen symbolischen Link für die Installation des Panels mithilfe von Parallels Containers EZ-Templates auf RPM-basierten Betriebssystemen. Falls Sie einen Mirror für eine Umgebung ohne Parallels Containers einrichten, lassen Sie diesen Schritt aus.

```
# cd destination_directory; ln -s PPSMBE_10.2.0 PPSMBE10
```

- Mithilfe dieses Befehls können Sie die Konfigurationsdateien des Parallels-Installers herunterladen.

```
# rsync -auv—delete
rsync://rsync.autoinstall.plesk.com/'autoinstall/products.inf3
autoinstall/versions.inf3  autoinstall/ppsmbe.inf3
autoinstall/setemplates.inf3' destination_directory
```

5. Bearbeiten Sie die Konfigurationsdateien des Parallels-Installers, um festzulegen, welche Produkte, Versionen und Plattformen vom Mirror unterstützt werden sollen.

   a. Bearbeiten Sie die Datei `products.inf3` und entfernen Sie alle **product**-Elemente bis auf die mit der ID `ppsmbe`. Lassen Sie auch das **product**-Element mit der ID `setemplates` aus, wenn das zusätzliche Set an Site-Design-Templates für Site Editor unterstützt werden sollen.

   b. Bearbeiten Sie die Datei `ppsmbe.inf3` und entfernen Sie alle **Build**-Elemente bis auf die für Betriebbssysteme und Architekturen, die vom Mirror unterstützt werden sollen.

   c. Bearbeiten Sie die Datei `setemplates.inf3` und entfernen Sie alle **Build**-Elemente bis auf die für Betriebbssysteme und Architekturen, für die zusätzliche Site-Design-Templates von Site Editor unterstützt werden sollen.

Hinweis: Weitere Informationen zu dem Dienstprogramm `Rsync` finden Sie unter http://samba.anu.edu.au/rsync/.

Mirrors von Windows-Distributionen erstellen

Das Repository von Parallels Small Business Panel für Windows befindet sich unter http://autoinstall-win.parallels.com und ist wie folgt strukturiert (es werden nur die Dateien und Verzeichnisse aufgelistet, die für das Erstellen von Mirrors der Panel-Pakete relevant sind):

- *<PRODUKTNAME>_<Produkt_Version>/*
  Mehrere Unterverzeichnisse werden in Übereinstimmung mit den Panel-Versionen genannt.
  In unserem Fall ist das Unterverzeichnis `PPSMBE-WIN_<Version>/` interessant, z.B., `PPSMBE-WIN_10.2.0/`
  Jedes dieser Unterverzeichnisse enthält die folgenden Dateien:
  - `dist-<Typ>-<Name-des-Betriebssystems>-<Version>-<Architektur>/`
    Enthält Distributionspakete des Panels oder die Design-Templates
    z.B., `dist-msi-Microsoft-2003-i386/`
  - `thirdparty-<Typ>-<Name-des-Betriebssystems>-<Version>-<Archiektur>`
    weitere Drittanbieter-Pakete
  - `ppsmbe-<Parallels-Panel-Version>-<Name-des-Betriebssystems>-<Architektur>.inf3`
    Parallels-Installer-Konfigurationsdatei
- `products.inf3`
  Die Parallels-Installer-Konfigurationsdatei, welche die Panel-Produkte beschreibt
- `ppsmbe.inf3`
  Parallels-Installer-Konfigurationsdateien, die unterschiedliche Versionen des Panels und weitere Komponenten beschreiben

➢ *So erstellen Sie einen Mirror für das Panel:*

1. Melden Sie sich an dem Server an, auf dem der Mirror eingerichtet werden soll.
2. Erstellen Sie ein Verzeichnis, in dem die Dateien abgelegt werden sollen. Zur Veranschaulichung nennen wir dieses Verzeichnis `destination_directory/`.
3. Führen Sie die folgenden Befehle aus, um einen Mirror einzurichten.
   - Mit diesem Befehl können Sie das Verzeichnis `PPSMBE-WIN_10.2.0` aus dem Parallels-Updates-Repository in das Verzeichnis `destination_directory` herunterladen.

```
# rsync -au—delete
rsync://rsync.autoinstall-win.parallels.com/autoinstall/PPSMBE-WIN
_10.2.0 destination_directory
```

   - Mithilfe diesen Befehls können Sie die Parallels-Installer-Konfigurationsdateien herunterladen.

```
# rsync -auv—delete
rsync://rsync.autoinstall-win.parallels.com/'autoinstall/products.
inf3 autoinstall/ppsmbe.inf3' destination_directory
```

4. Bearbeiten Sie die Parallels-Installer-Konfigurationsdateien, um festzulegen, welche Produkte, Versionen und Plattformen vom Mirror unterstützt werden sollen.

   a. Bearbeiten Sie die Datei `products.inf3` und entfernen Sie alle **product**-Elemente bis auf die mit der ID `ppsmbe`. Lassen Sie auch das **product**-Element mit der ID `setemplates` aus, wenn das zusätzliche Set an Site-Design-Templates für Site Editor unterstützt werden sollen.

   b. Bearbeiten Sie die Datei `ppsmbe.inf3` und entfernen Sie alle **Build**-Elemente bis auf die für Betriebbssysteme und Architekturen, die vom Mirror unterstützt werden sollen.

Hinweis: Weitere Informationen zu dem Dienstprogramm `Rsync` finden Sie unter http://samba.anu.edu.au/rsync/.

KAPITEL 4

Überblick zu Setupvorgängen

In diesem Kapitel erhalten Sie einen Überblick über die wichtigsten Setup-Vorgänge während der Bereitstellung des Parallels Small Business Panel.

Die Beschreibungen der Vorgänge in diesem Kapitel werden in der Reihenfolge aufgeführt, wie sie am besten durchgeführt werden sollten.

In diesem Kapitel:

Initialisierung des Panels 34
Installation des Lizenzschlüssels 35
Branding des Panels 36
Anpassen von Service-Links 36
DNS-Konfiguration 40
Installation von SSL-Zertifikaten 42
Erstellen von Domains und Subdomains 43
Installieren von Anwendungen 45
Verfügbare APS-Kataloge definieren 47

Initialisierung des Panels

Nach der Installation des Panels muss es im ersten Schritt initialisiert werden. Es ist nicht zulässig, andere Operationen - weder in der Benutzeroberfläche des Panels noch über die API - auszuführen.

Während der Initialisierung werden folgende Panel-Einstellungen konfiguriert:

- E-Mail-Adresse des Administrators
- das neue Passwort des Administrators (anstelle des standardmäßig eingerichteten Passworts)
- (optional) Panel-Sprache

  Wenn während der Initialisierung des Panels keine Spracheinstellungen konfiguriert werdem, wird die standardmäßig voreingestellte Sprache “American English (en-US)” verwendet.

  Die Sprache der Benutzeroberfläche kann zu einem späteren Zeitpunkt vom Panel-Administrator jederzeit geändert werden (vorausgesetzt die benötigte Sprachkomponente (auf Seite 12) wurde installiert).

  Falls Sie eine benutzerdefinierte Lokalisierung des Panels bereitstellen, müssen Sie ein benutzerdefiniertes Sprachpaket vor der Initialisierung des Panels importieren. Weitere Informationen finden Sie in der *Lokalisierungsanleitung für Parallels Small Business Panel*, welches auf der Dokumentationsseite des Panels (http://www.parallels.com/products/small-business-panel/documentation/) zur Verfügung steht.

Quicklinks

Initialisierung des Panels über Remote-API (auf Seite 89)

Initialisierung des Panel über die Befehlszeilenoberfläche (CLI) (auf Seite 210)

Installation des Lizenzschlüssels

Das Panel wird mit einem standardmäßig voreingestellten Lizenzschlüssel vertrieben, der eine begrenzte Anzahl an Benutzer-Accounts unterstützt. Aus diesem Grund ist es notwendig, einen Lizenzschlüssel zu erwerben und zu installieren, der am besten die Anforderungen des jeweiligen Benutzers erfüllt.

Zusammen mit dem Hauptlizenzschlüssel, der die wichtigsten Funktionalitäten des Panels definiert, können zusätzliche Lizenzschlüssel installiert werden, welche die Nutzung von Panel-Add-ons (z.B. Parallels Premium Antivirus) ermöglichen.

Quicklinks

Installation des Lizenzschlüssels über Remote-API (auf Seite 94)

Installation des Lizenzschlüssels über die Befehlszeilenoberfläche (CLI) (auf Seite 211)

Branding des Panels

Die im Folgenden genannten Branding-Aspekte des Panels fallen in den Bereich der Panel-Bereitstellung:

- Panel-Firmenzeichen und URL
- Support-Button (Link)

Hinweis: Beide werden nur in der CLI unterstützt.

Logo

Das Panel-Logo wird in der linken oberen Ecke des Panels angezeigt. Standardmäßig handelt es sich um eine Bilddatei im GIF-Format (141 x 29 Pixel) mit dem Produktlogo von Parallels Small Business Panel. Wenn Sie auf dieses Logo klicken, öffnet sich die Website von Parallels. Sie können das standardmäßig voreingestellte Logo durch Ihr eigenes und der URL zu Ihrer Firmen-Website ersetzen.

Support

Der **Support**-Button befindet sich auf der Registerkarte **Einstellungen** und ist für den Panel-Administrator verfügbar. Beim Klicken auf diesen Button, öffnet sich standardmäßig das Support-Anfrageformular der Parallels-Website, welches vom Administrator ausgefüllt und versendet wird.

Falls Sie Ihren Kunden technischen Support anbieten möchten, können Sie die URL verändern, die sich öffnet, wenn man auf den **Support**-Button klickt.

Quicklinks

Branding des Panels über die Befehlszeilenoberfläche (CLI) (auf Seite 211)

Anpassen von Service-Links

Die Benutzeroberfläche von Parallels Small Business Panel beinhaltet verschiedene Links zu den folgenden Services:

Dienst	Beschreibung	Zugriff von	Standard
Online-Shop des Panels	Verkauf/Upgrade von Panel-Add-ons und allgemeinen Lizenzen.	**Einstellungen > Verwaltung von Lizenzschlüsseln**: die Links **Online-Shop besuchen und Panel-Add-ons erwerben** und **Online-Shop besuchen und Panel-Lizenzschlüssel upgraden**	Online-Shop von Parallels

Marketplace Application Online-Shop (bitte lesen Sie den **Hinweis** unter der Tabelle)	Lizenzen-Verkauf für Marketplace-	**Applikationen & Skripte > Marketplace**: die **Jetzt kaufen**-Buttons	Parallels Marketplace-Shop
Shop für Benutzerpakete	Verkauf von Panel-	▪ **Benutzer > Benutzer-Accounts**: der Link **Zusätzliches Benutzerpaket kaufen** ▪ **Assistent "Erste Schritte" > Benutzer**: der Link **Zusätzliches Benutzerpaket kaufen** ▪ **E-Mail > E-Mail-Adressen**: der Link **Zusätzliches Benutzerpaket kaufen** ▪ **Einstellungen > Verwaltung von Lizenzschlüsseln**: der Link **Marketplace besuchen und Benutzerpakete kaufen**	Parallels Marketplace-Shop
Online-Shop für	Verkauf von CA SSL-Zertifikate.	**Einstellungen > SSL-Zertifikate > SSL-Zertifikate hinzufügen**: der Button **SSL-Zertifikate kaufen**	myplesk.com
Domain Name Registrar	Verkauf von Domain-Namen.	**Websites & Domains**: der Button **Domain-Namen registrieren**	myplesk.com

Hinweis: Bitte beachten Sie den Unterschied:*Marketplace Application Catalog* ist ein Service, der eine detaillierte Liste mit den verfügbaren Anwendungen bereitstellt. Der *Marketplace Online Shop for Applications* ist ein Service über den Lizenzen für diese Anwendungen verkauft werden.
Die URLs von Anwendungskatalogen (inkl. Application Marketplace) können individuell angepasst werden, siehe hierzu auch den Abschnitt **Verfügbare APS-Kataloge definieren** (auf Seite 47).

In den folgenden Fällen sollten Sie die URLs verändern, die sich beim Anklicken öffnen:

- Sie verkaufen als Parallels-Partner Lizenzen für das Panel und die Add-ons
- Sie verkaufen als Parallels-Partner Benutzerpakete für das Panel
- Sie verkaufen als Parallels-Partner Lizenzen für Anwendungen, die über das Marketplace erhältlich sind
- Sie bieten Registrierungsdienste für Domain-Namen an
- Sie verkaufen CA-Zertifikate

Sie können die Links anpassen, indem Sie die Konfigurationsdatei des Panels `smb.ini` entsprechend bearbeiten. Sobald die erforderliche Anpassung in der Konfigurationsdatei vorgenommen wurde, kann sie auf eine beliebige Anzahl von Panel-Instanzen hochgeladen werden und so die standardmäßig voreingestellte Konfiguration ersetzen. Hierzu sind keine besonderen Mittel notwendig, bitte verwenden Sie daher die Standardbefehle.

Die Konfigurationsdatei des Panels

Speicherort:

- Auf Linux/Unix: `$PRODUCT_ROOT/admin/smb/application/config/smb.ini`
- Auf Windows: `%plesk_dir%\admin\smb\application\config\smb.ini (%plesk_dir% ist eine Umgebungsvariable, die das Panel-Installationsverzeichnis angibt)`

Konfiguration von Service-Links:

```
[marketplace]
panelAndAddonsLicensesStore = "http://example.com/panel-store"
url =  "http://example.com/marketplace-shop"
userpacksUrl = "http://example.com/panel-userpacks"
buySslCertificatesUrl = "http://example.com/certificates"
registerDomainsUrl = "http://example.com/certificates"
```

Standardmäßig beinhaltet die Konfigurationsdatei diesen Abschnitt nicht, daher müssen Sie manuell den Abschnittstitel hinzufügen (`[marketplace]`) und die folgenden Strings, je nachdem, welche Funktionen angepasst werden soll.

- falls die URL des Online-Shops geändert werden soll, über den das Panel und die Add-ons verkauft werden:

  ```
  panelAndAddonsLicensesStore = "http://example.com/panel-store"
  ```

- falls die URL des Online-Shops geändert werden soll, über den Lizenzen für Marketplace-Anwendungen verkauft werden:

```
url = "http://example.com/marketplace-shop"
```

- falls die URL des Online-Shops geändert werden soll, über den die Benutzerpakete für das Panel verkauft werden:

```
userpacksUrl = "http://example.com/panel-userpacks"
```

- falls die URL geändert werden soll, über die CA-Zertifikate verkauft werden:

```
buySslCertificatesUrl = "http://example.com/certificates"
```

- falls die URL geändert werden soll, über die Domain-Namen registriert werden:

```
registerDomainsUrl = "http://example.com/certificates"
```

Um Änderungen rückgängig zu machen und zu den Standardeinstellungen zurückzukehren, entfernen Sie den entsprechenden String aus `smb.ini`.

DNS-Konfiguration

<SMB_panel> wird zusammen mit dem DNS-Server installiert. Dieser Server wurde als der Master-Nameserver für die Domains konfiguriert, die mit dem Panel erstellt und gehostet werden. Bei der Bereitstellung des Panels können Sie den DNS-Service nach Bedarf konfigurieren oder deaktivieren.

Falls Sie eine Arbeitsumgebung mit einem Nameserver eingerichtet haben, der alle gehosteten Domains verwaltet, müssen Sie höchstwahrscheinlich keinen separaten DNS-Server innerhalb eines jeden Servers mit einer Panel-Instanz einrichten. Oder anders formuliert: Sie müssen höchstwahrscheinlich nicht für jede Panel-Installation einen eigenen DNS-Sever anlegen, sondern können das Panel an einen bestehenden DNS-Server anbinden. Es sollte auch erwähnt werden, dass im Fall einer Parallels Containers-Umgebung eventuell mehr Ressourcen verbraucht werden. Denn Container mit dem Panel, auf denen der DNS-Server ausgeführt wird, verbrauchen mehr Ressourcen, wodurch unter Umständen die Dichte der gesamten Hosting-Umgebung geringfügig gesenkt wird.

Um DNS-Zonen von Panel-verwalteten Domains so zu konfigurieren, dass diese von einem Remote-DNS-Server beantwortet werden, müssen Sie den DNS-Service bei der Bereitstellung der Panel-Instanz deaktivieren. Falls Ihre Kunden sich entschliessen, dass Ihre Domains von einem lokalen DNS-Server verarbeitet werden sollen, können Sie diesen später immer noch aktivieren.

Falls die vom Panel verwalteten Domains von einem lokalen DNS-Service verarbeitet werden sollen, können Sie die DNS-Zonen jederzeit nach Bedarf verändern.

Das Panel stellt ein anpassbares Template für DNS-Zonen zur Verfügung, das Templates für DNS SOA- und andere Ressourceneinträge enthält. Wenn ein Domain-Account erstellt wurde, wird die DNS-Zone basierend auf diesem Template erstellt. Um DNS-Zonen für die vom Panel verwalteten Domains zu erstellen, müssen Sie nur das entsprechende DNS-Zonen-Template bearbeiten.

Quicklinks

DNS-Konfiguration per Remote-API (auf Seite 100)

DNS-Konfiguration per CLI (auf Seite 213)

In diesem Abschnitt:

Ändern des SOA-Record-Templates 41
Einrichten von Templates für Ressourceneinträge (Resource Records) 41

Ändern des SOA-Record-Templates

Sie können in dem Panel-Template für den SOA-Record folgende Eintragsfelder verändern:

- `ttl (Time to Live)`
- `Refresh/Aktualisieren`
- `Retry/Wiederholen`
- `Expire/Ablauf`
- `Minimum`
- (nur Unix-CLI) `sn` (= Seriennummer) Format
  Format der Zonen-Seriennummer: yyyymmdd oder Unix Timestamp

Einrichten von Templates für Ressourceneinträge (Resource Records)

Das Panel-Template der DNS-Zone unterstützt die folgenden Ressourceneintragstypen:

- A
- NS
- CNAME
- MX
- PTR
- TXT
- AXFR
- SRV

Installation von SSL-Zertifikaten

Um ein Secure Sockets Layer-Protokoll für eine Domain zu übernehmen, muss zuerst ein SSL-Zertifikat installiert werden.

In dem Panel werden SSL-Zertifikate mit Domains über IP-Adressen verbunden: ein Zertifikat wird einer IP-Adresse zugewiesen und dieses Zertifikat wird verwendet, wenn auf eine Domain zugegriffen wird, die auf dieser IP gehostet wird. In der Tat wird das Zertifikat verwendet, wenn versucht wird, über https auf alle Domains zuzugreifen, die auf dieser IP gehostet werden.

Um ein SSL-Zertifikat zu installieren, müssen Sie ein Zertifikat in das Panel importieren und einer IP-Adresse zuweisen. Wenn ein Domain-Account auf einer solchen IP erstellt wird, wird der Domain-Hostingparameter “SSL-Unterstützung” automatisch aktiviert und das installierte Zertifikat wird verwendet, um auf diese Domain zuzugreifen.

Quicklinks

Installation von SSL-Zertifikaten über Remote-API (auf Seite 128)

Installation von SSL-Zertifikaten über CLI (auf Seite 220)

Erstellen von Domains und Subdomains

Domains

Der Erstellungs- und Konfigurationsprozess von Domains und Subdomains wurde in Parallels Small Business Panel neu eingerichtet und so einfach wie möglich gestaltet. Die Idee ist, dass Sie bei der Erstellung einer Domain oder einer Subdomain die grundlegenden Hostingparameter angeben und der Rest der Hostingeinstellungen wird automatisch angepasst.

Um eine Domain zu erstellen, geben Sie die folgenden Argumente an:

- Domainname.
- IP-Adresse.

  Falls die IP-Adresse mit einem SSL-Zertifikat verknüpft ist (auf Seite 42), dann wird die SSL-Unterstützung automatisch für die Domain auf dieser IP aktiviert.
- Erforderliche Webhosting-Eigenschaften: Login und Passwort des FTP-Accounts, der für das Hochladen des Websiten-Inhalts verwendet wird.

  Wenn eine Webanwendung auf einer Domain installiert wird, richtet das Panel automatisch alle für diese Anwendung benötigten Dienste ein.Aus diesem Grund sind die IP-Adresse, FTP-Login und Passwort die einzigen Hostingeinstellungen, die während der Erstellung der Domain angegeben werden müssen.

Quicklinks

Domain-Erstellung über Remote-API (auf Seite 135)

Domain-Erstellung über CLI (auf Seite 222)

Subdomains

Wenn Sie eine Subdomain erstellen, geben Sie mindestens den Präfixnamen der Subdomain und die übergeordnete Domain an.

Der Inhalt einer Subdomain befindet sich normalerweise in dem Verzeichnis `/httpdocs/subdomains/`*`subdomain-name`* in dem virtuellen Host der Domain.

Das Panel für Windows unterstützt auch das so genannte *Subdomains auf Unterordner* (Subdomains on Subfolder) - Subdomains gemappt auf ein beliebiges Verzeichnis, das sich in dem Ordner `/httpdocs` befindet. Das ist besonders nützlich, wenn eine installierte Webanwendung als Subdomain zugänglich sein soll. Beispiel: Eine WordPress-Anwendung ist normalerweise über URLs wie z.B. `http://domain-name/URL-prefix` beispielsweise `http://example.com/WordPress` erreichbar. Mit einer Subdomain auf einem Unterordner ist es auch möglich, die Anwendung über `http://blog.example.com` zur Verfügung zu stellen.

- Um eine reguläre Subdomain zu erstellen, geben Sie den Namen der übergeordneten Domain, den Namen der Subdomain und die Hostingeinstellungen der Subdomain an.

  Hinweis: Standardmäßig ist der Subdomain-Inhalt unter demselben FTP-Account wie den der übergeordneten Domain verfügbar. Dementsprechend ist die Angabe der FTP-Account-Einstellungen optional und wird nur vorgenommen, wenn ein separater FTP-Account erforderlich ist, um auf den Inhalt einer Subdomain zu zugreifen.

- Um eine Subdomain auf einem Unterordner zu erstellen (nur Windows-Unterstützung), geben Sie den Namen der übergeordneten Domain/Subdomain, den Subdomain-Namen und den Pfad zu einem Ordner an, auf dem die Subdomain erstellt wird.

  Hinweis: Für Subdomains auf einem Unterordner ist die Erstellung eines separaten FTP-Benutzer-Accounts nicht möglich: es wird immer der FTP-Benutzer-Account der übergeordneten Domain/Subdomain verwendet.

Quicklinks

Erstellung von Subdomains über API (auf Seite 141)

Erstellung von Subdomains über CLI (auf Seite 223)

Installieren von Anwendungen

Parallels Small Business Panel unterstützt APS-Applikationen. Dabei handelt es sich um Webanwendungen, die gemäß dem APS - Application Packaging Standard (http://www.apsstandard.org/) verpackt werden, wodurch das SaaS-Modell ins Hosting-Business integriert werden kann.

Wir unterscheiden den Term *Anwendungspaket* - ein Zip-Archiv mit einer Webanwendung, die in dem APS-Format verpackt wurde. Im Gegensatz zu der Anwendung an sich, die von dem Paket aus installiert wird.

Für die Installation einer Anwendung auf einer Domain oder Subdomain sind zwei Schritte notwendig:

1. Erhalt des Anwendungspakets und der Import in das Panel.
2. Installation der Anwendung von dem Paket auf eine Domain oder eine Subdomain.

Quicklinks

Installation von Anwendungen über Remote-API (auf Seite 150)

Installation von Anwendungen über CLI (auf Seite 225)

In diesem Abschnitt:

Importieren eines Anwendungspakets 46
Installieren einer Anwendung 46

Importieren eines Anwendungspakets

Sie können zwischen zwei Vorgehensweisen wählen, um ein Anwendungspaket abzurufen und zu importieren:

- Ein Paket, wahrscheinlich ein selbst erstelltes, wird manuell auf den Host, auf dem das Panel installiert wurde, hochgeladen und dann manuell in das Panel importiert.
- Ein Paket wird heruntergeladen und automatisch aus einem APS-Katalog in das Panel importiert (ein Webdienst bietet Zugang zu Remote gespeicherten Anwendungspaketen).

  Es ist möglich, mehrere Anwendungen auf einmal herunterzuladen und zu importieren. Das Panel unterstützt die Nachverfolgung des Download-Status.

Das Panel ermöglicht es zudem, Anwendungspakete über mehrere APS-Katalog-Instanzen abzurufen. Verfügbare APS-Kataloge werden in der Konfigurationsdatei des APS-Katalogs (auf Seite 47) definiert, die an die Panel-Instanz in einer bestimmten Hostingumgebung angepasst werden können.

Installieren einer Anwendung

Wenn eine Anwendung installiert wird, sollten die folgenden Angaben spezifiziert werden:

- Anwendungspaket, aus der die Anwendung installiert werden soll.

  Identifiziert entweder durch die Information “name-version-release” oder durch die Paket-ID. Diese Daten erhalten Sie, wenn Sie die verfügbaren Pakete abrufen. Wenn Sie die verfügbaren Pakete abrufen, erhalten Sie eine detaillierte Liste mit den Anwendungspaketen, die in das Panel importiert werden sowie den benötigten IDs.
- Ziel-Domain oder Subdomain.
- Installationseinstellungen.

  Diese sind anwendungsspezifisch; es können auch Anwendungen ohne Installationseigenschaften oder Einstellungen vorhanden sein. Die Panel-API unterstützt nicht die Informationsabfrage von Anwendungseinstellungen. Sie sollten diese abfragen, indem Sie die API des APS-Katalogs (http://www.apsstandard.org/r/doc/aps-catalog-1.1-api/index.htm) verwenden oder indem Sie die Metadaten-Datei des Pakets parsen. Eine Beschreibung des APS-Formats finden Sie in den APS-Format-Spezifikationen unter http://www.apsstandard.org/providers/documentation/.

Verfügbare APS-Kataloge definieren

Die Liste mit den APS-Katalogen, die in der Konfigurationsdatei definiert wird, wird aus den folgenden zwei Gründen angegeben:

1. Anwendungen aus diesen Katalogen können direkt über die Panel-API auf einer Domain installiert werden.

   Hinweis: Es ist nicht möglich, Anwendungen aus einem APS-Katalog zu installieren, die nicht in der Konfigurationsdatei aufgelistet werden.

2. Diese Kataloge werden dem Panel-Benutzer auf dem Bildschirm **Applikationen & Skripte** angezeigt und Benutzer können die Anwendungen selbst installieren.

APS-Katalog ist der allgemeine Name des Dienstes, der auf folgende Weise angegeben werden kann:

- APS-Katalog unter apscatalog.com (http://apscatalog.com). Offizieller APS-Katalog, der von Parallels unterstützt wird.

  Hinweis: Es ist wichtig, dass apscatalog.com in der Konfiguration angegeben wird, wenn Websites mit APS-Anwendungen zu dem Panel Backup/Restore migriert werden.

- Benutzerdefinierter APS-Katalog, ein lokaler Mirror des Dienstes, der innerhalb der Hostingumgebung bereitgestellt wurde und Zugang zu bestimmten Anwendungen bietet.

Weitere Informationen darüber, wie Sie einen lokalen Mirror eines APS-Katalogs implementieren, finden Sie in der Dokumentation zum APS-Standard. Diese Dokumentation können Sie sich über die offizielle Website http://www.apsstandard.org/providers/documentation/ herunterladen.

In den folgenden Fällen sollten Sie verfügbare APS-Kataloge definieren:

a Bei der Bereitstellung des Panels, können Sie Anwendungen von Ihrem benutzerdefinierten APS-Katalog aus installieren.

Fügen Sie in diesem Fall einen Eintrag für Ihren Katalog zu der Konfigurationsdatei hinzu.

b Sie wollen, dass Panel-Benutzer nur auf Ihren benutzerdefinierten APS-Katalog zugreifen oder parallel zu denen, die vom Panel standardmäßig bereitgestellt werden.

In diesem Fall können Sie einen Eintrag für Ihren Katalog zur Konfigurationsdatei hinzufügen und optional bestehende Einträge entfernen.

c Panel-Benutzer sollen überhaupt keinen Zugang zu einem der APS-Kataloge haben.

In diesem Fall entfernen Sie alle Einträge aus der Konfigurationsdatei.

Um die Liste mit den verfügbaren APS-Katalogen zu bearbeiten, sollten Sie die folgenden Vorgänge durchführen:

1. Passen Sie die Beispiel-APS-Kataloge (auf Seite 49) an Ihre Anforderungen an.
2. Wählen Sie eine der folgenden Vorgehensweisen, um die Konfigurationsdatei in das Panel zu importieren:

- Kopieren Sie die benutzerdefinierte Konfigurationsdatei und ersetzen Sie die Standarddatei durch Standardbetriebssystem-Methoden
- Import über die Panel-Bereitstellungs-API

Quicklinks

Definieren von verfügbaren APS-Katalogen über Remote-API (auf Seite 146)

Definieren von verfügbaren APS-Katalogen über CLI (auf Seite 224)

In diesem Abschnitt:

Konfigurationsdatei von APS-Katalogen 49

Konfigurationsdatei von APS-Katalogen

Speicherort

Auf Linux/Unix:
`/usr/local/psa/admin/smb/application/config/aps/catalogs.ini`

Auf Windows: `%plesk_dir%\admin\smb\application\config\aps\catalogs.ini`

Format

Die Konfigurationsdatei von APS-Katalogen ist eine Textdatei in dem folgenden Format:

```
[apscatalog]
;Key word serving as internal identifier of the APS applications
resource.
type = apscatalog
;APS applications resource type definition, can be marketplace or
apscatalog
;marketplace is a type used to denote Application Marketplace service
```

;apscatalog is a common type that should be used for any APS Catalog (official APS Catalog service and local mirrors)

url = http://apscatalog.com

;APS applications resource URL

[marketplace]

type = marketplace

url = http://catalog.marketplace.parallels.com

protocol = all-app

[APSCatalogLocalMirror]

type = apscatalog

name.en-US = “APS Catalog (local)”

;APS applications resource title as it will be displayed in the Panel GUI in the corresponding locale

description.en-US = “Local instance of APS catalog”

;APS applications resource description as it will be displayed in the Panel GUI in the corresponding locale

url = apscatalog.example.com

KAPITEL 5

Einrichten über Remote-API

Dieses Kapitel konzentriert sich auf das Einrichten des Panels mithilfe der Remote-API des Panels (API, Abk. für Application Programming Interface, deutsch: Programmierschnittstelle).

Der Abschnitt **Remote-API verwenden** (auf Seite 51) beinhaltet einen Überblick über das API RPC-Protokoll und Tool-Samples, die verwendet werden, um mit einem vom Panel verwalteten Server über das Protokoll zu kommunizieren.

Der Abschnitt **Einrichten des Panels** (auf Seite 89) beschreibt im Detail Abfragenachrichten, die ein Drittanbieter-Tool ausgeben muss, um Panel-Setup-Operationen sowie Antwortnachrichten, die nach jeder durchgeführten Operation empfangen werden, auszuführen.

In diesem Kapitel:

Remote-API verwenden 51
Einrichten des Panels..... 89

Remote-API verwenden

Dieser Abschnitt beschreibt, wodurch sich die Remote-API des Panels auszeichnet, erklärt die Funktionsweise und wie eine Client-Anwendung erstellt wird, die mit dem Panel über Remote-API kommuniziert.

In diesem Abschnitt:

Über Remote-API 52
API RPC-Pakete 53
Erstellen der Client-Software 57

Über Remote-API

Um die Kommunikation zwischen dem Panel und einer Drittanbieter-Software zu gewährleisten, stellt das Panel eine XML-basierte API (API, Abk. für Application Programming Interface, deutsch: Programmierschnittstelle) bereit. Diese Schnittstelle macht eine Reihe an Funktionen für die Verwaltung von logischen Objekten des Panels verfügbar. Das API RPC-Protokoll wurde entwickelt, um “remote” d.h. von einem entfernten Standort aus mit diesen API-Funktionen zu kommunizieren. Es handelt sich um das “XML-over-HTTP Protocol”, das Daten in Form von speziell formatierten Paketen austauscht.

Die Kommunikation zwischen dem Panel und einer Drittanbieter-Software basiert auf dem Client/Server-Programmierungsmodell.

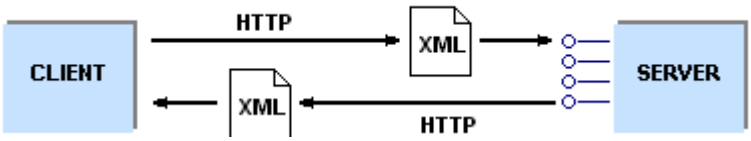


Рисунок 1: Kommunikation zwischen Parallels Panel und einer Drittanbieter-Client-Software

Der Server ist in dem Fall der Panel API RPC-Server (nachfolgend *der Server*), der ein Bestandteil des Panels ist. Die Client-Software sollte von einem Drittanbieter stammen. Das Kommunikationsszenario stellt sich wie folgt dar:

1. Die Client-Anwendung erstellt ein XML-basiertes Abfrage-Paket (nachfolgend *API RPC-Paket* oder *das Paket*), fügt einen Standard-HTTP-Header hinzu und sendet diesen an den Server.
2. Der Server empfängt das Paket, validiert dieses und ruft interne Panel-Funktionen auf, um die angeforderten Operationen durchzuführen.
3. Der Server erstellt das Antwortpaket, das die notwendigen Informationen zu der durchgeführten Operation enthält und sendet es zurück.
4. Der Client erhält dieses Paket (mit dem Status zum Operationsstatus und (eventuell) eine Fehlermeldung oder die Operationsparameter) und ruft diese Informationen ab.

   Die Kommunikation endet.

Unterstützte API-Versionen

Die aktuelle Version des Panels unterstützt die API RPC-Protokollversion 1.6.2.0.

API RPC-Pakete

Ein API RPC-Paket ist ein einfaches XML-Dokument, das in einem HTTP-Nachrichtentext enthalten ist.

Wir unterscheiden die folgenden zwei Pakettypen:

- *Abfrage-Pakete* - werden von der Client-Software erstellt und an den Server via Remote-API gesendet
- *Antwortpakete* - werden von dem Server erstellt und an den Client gesendet
  Ein Antwortpaket enthält den Ausführungsstatus der angeforderten Operation und kann eine Fehlermeldung enthalten, wenn die Operation nicht ausgeführt werden konnte.

In diesem Abschnitt:

Paketstruktur 54
Beispiel für ein Paket 54
Validierung von Paketen 56

Paketstruktur

Wir unterscheiden die folgenden API RPC-Paketteile:

1. *HTTP-Header* - Transportteil. Neben anderen Standardinformationen muss dieser Teil Daten zu dem Server enthalten, auf dem die Operation ausgeführt wird und die Anmeldeinformationen des Panel-Administrators.
2. *Paket-Header* - Angabe der API RPC-Protokollversion.
3. *Paketkörper* - Definition der Operationen, die auf dem Server durchgeführt werden sollen.

Der Paketkörper ist wie folgt strukturiert:

<[operator]>

<[operation]>

```
  <[parameter1]>…</[parameter1]>
...
  <[parameter2]>…</[parameter2]>
      …
```

</[operation]>

</[operator]>

Dieses Beispiel zeigt die *allgemeine* Struktur des Paketkörpers, mit einem *[operator]*-Element und einem *[operation]*-Element. Die Bezeichnungen *[operator]*, *[pperation]*, *[parameter]* sollten durch echte Operatoren, Funktionen und Parameter ersetzt werden.

Hinweis: Innerhalb eines Operators können mehrere Operationsbereiche existieren und viele Operatorbereiche innerhalb eines einzigen Pakets.

Im Nachfolgenden werden XML-Paket-Header und Körper zusammen als *Paket* bezeichnet.

Beispiel für ein Paket

Hier finden Sie ein Beispiel für eine API RPC-Abfragenachricht an den Server:

`POST /enterprise/control/agent.php HTTP/1.1`
`Host: 10.58.83.1:8443`
`Accept: /`
`HTTP_AUTH_LOGIN: admin`
`HTTP_AUTH_PASSWD: Setup`
`Pragma: no-cache`
`Content-Length: 1398`
`Content-Type: text/xml`
	HTTP-Header
	Der HTTP-Header weist darauf hin, dass POST als HTTP-Methode verwendet wird. Die Benutzerschnittstelle befindet sich unter `/enterprise/control/agent.php` und die HTTP-Version ist 1.1. Das `Host`-Element gibt die IP-Adresse und den Port des Panel-Servers an, an den die Nachricht gesendet wurde. Die Elemente `HTTP_AUTH_LOGIN` und `HTTP_AUTH_PASSWD` beinhalten die Anmeldeinformationen des Panel-Administrators.Nach `Content-Type` muss “text/xml” stehen. Die Länge der weitergeleiteten Nachricht ist auch erforderlich.

`<?xml version="1.0" encoding="UTF-8" standalone="no"?>`	**Anfang des XML-Teils**
`<packet version="1.6.2.0">`	**Paket-Header** Das Attribut *Version* gibt die benötigte Version des Panel-API RPC-Protokolls an.

`<smb>`
`<initial_setup>`
`<password>P4$$w0rd</password>`
`<admin_email>johndoe@example.com</admin_email>`
`<locale>de-DE</locale>`
`</initial_setup>`
`</smb>`
	Paketkörper
	Beginnt immer mit dem Tag des zugehörigen Operators.
	Dieses Beispiel-Paket verwendet den **smb**-Operator, um eine kürzlich installierte Panel-Software zu initialisieren. Die innerhalb des Knotens von **initial_setup** geschachtelten Elemente enthalten die Daten, die für die Initialisierung des Panels notwendig sind: E-Mail-Adresse und Passwort des Panel-Administrators und die Panel-Sprache.

`</packet>`	**Abschließender Tag des Pakets**

Sowohl Abfrage- und Antwortpakete sind sich ähnlich in der Struktur, aber der HTTP-Header und Körper unterscheiden sich. Die Nachricht, die der Server als Antwort auf die Abfrage versendet, könnte wie folgt aussehen:

`HTTP/1.1 200 OK`
`Transfer-Encoding: chunked`
`Content-type: text/html`
`Date: Fri, 03 Apr 2009 18:54:56 GMT`
`Server: sw-cp-server/1.0.0`
	Standard-Header der HTTP-Server-Antwort
	Der MIME-Typ ist text/xml. Das `Server`-Element zeigt, dass die Abfrage von dem Panel auf der Serverseite verarbeitet wurde.

`<?xml version="1.0" encoding="UTF-8" standalone="no"?>`	**Standard-Header eines gültigen XML-Dokuments**
`<packet version="1.6.2.0">`	**XML-Paket-Header**

`<smb>`
`<initial_setup>`
`<result>`
`<status>ok</status>`
`</result>`
`</initial_setup>`
`</smb>`
	Paketkörper
	Der Root-Knoten des Paketkörpers ist der **smb**-Operator.
	Die eingeschachtelten Zeilen weisen darauf hin, dass die Panel-Initialisierung erfolgreich war.

`</packet>`	**Der abschließende Tag des Pakets**

Validierung von Paketen

Der Panel API RPC-Server besteht aus einer Reihe von “Agents” - oder Operatoren - die vom Agent Engine verwaltet werden. Jeder Agent ist ein logischer Block, der für die Verarbeitung einer bestimmten Befehlsgruppe verantwortlich ist und der Agent Engine dient als “Paketverteiler”.

Desweiteren wird der Agent Engine mit verschiedenen Sets von XML-Schemadateien assoziiert. Jedes Set berichtet an eine bestimmte API RPC-Version. Ein Einsprungspunkt zu jeder Version ist die `agent_input.xsd`-Datei, die alle Eingangsschemata derselben Version referenziert.

Der Panel API RPC-Server funktioniert wie folgt: Zuerst berichtet das reine XML-Paket (ohne HTTP-Header) an den Agent Engine. Der Agent Engine ruft die API RPC-Version von dem Paket-Header ab, wählt das entsprechende Set an XML-Schmata aus und wechselt in das richtige `agent_input.xsd`-Schema. Dieses Schema validiert das Operator-Level des Pakets, nachdem jeder Operator-Abschnitt des Pakets mit dem passenden Folgeschema validiert wurde.

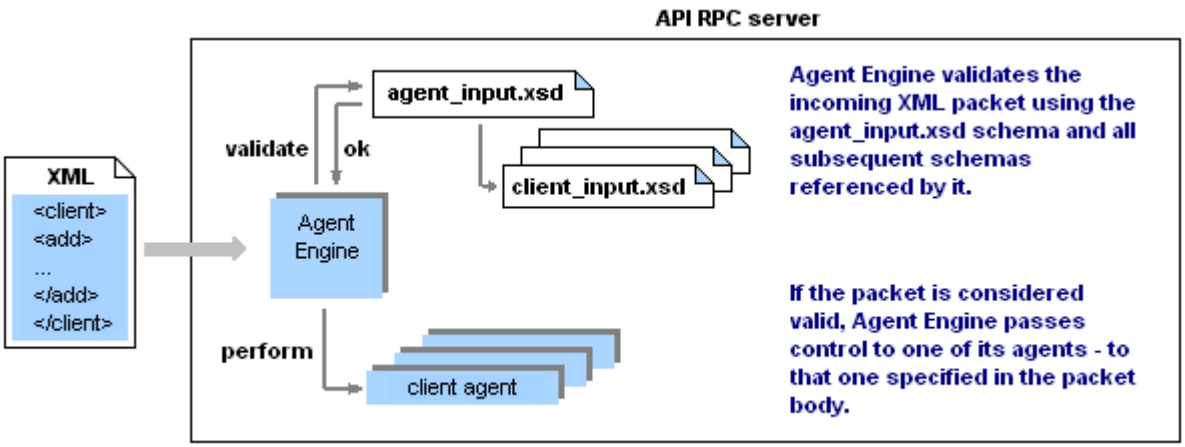

API RPC server
agent_input.xsd
validate
ok
client_input.xsd
XML
<client>
<add>
...
</add>
</client>
Agent Engine
perform
client agent
Agent Engine validates the incoming XML packet using the agent_input.xsd schema and all subsequent schemas referenced by it.
If the packet is considered valid, Agent Engine passes control to one of its agents - to that one specified in the packet body.

Рисунок 2: Validierung des Pakets auf der Serverseite

Wenn das Paket als ungültig analysiert wird, erstellt der Agent Engine einen Fehlerbericht und sendet diesen zurück an den Client. Wenn alle Elemente des Pakets erfolgreich validiert wurden, prüft der Agent Engine, ob alle notwendigen Module zur Weiterverarbeitung enthalten sind und ruft diese eines nach dem anderen ab.

Jeder Agent (Operator) empfängt seinen Abschnitt des Operator-Pakets, analysiert diesen, ruft interne Funktionen ab und liefert dem Agent Engine das Ergebnis der durchgeführten Befehle.

Erstellen der Client-Software

Die Client-Anwendung muss die folgenden Phasen des Datenaustausches umsetzen:

- Konvertierung des angeforderten Befehls in ein API RPC-aktiviertes XML-Paket
- Vorbereitung des Pakets zur Bereitstellung über HTTP und Weiterleitung an das Panel
- Antwort des Panels empfangen und die resultierenden Informationen aus dem Paket extrahieren

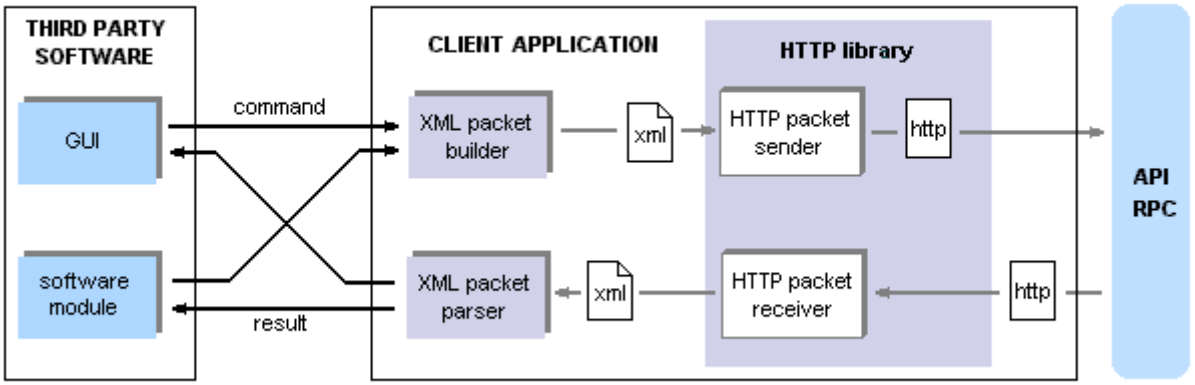


Рисунок 3: Struktur von clientseitigen Anwendungen

Die Abbildung zeigt, wie eine Client-Anwendung strukturiert werden kann und wie diese Struktur die Datenverarbeitung beeinflusst. Jedes Modul dieser Beispiel-Anwendung ist für einen bestimmten Task verantwortlich, d.h.:

1. Der *XML-Paket-Generator* akzeptiert den Befehl von der Anwendungs-GUI oder eines Drittanbieter-Moduls. Der XML-Paket-Generator enthält einen “Command-to-XML”-Konverter für jeden bekannten Befehl. Das Format des resultierenden XML-Pakets stimmt mit der API RPC-Version überein, die von der Client-Anwendung unterstützt wird.
2. Der *HTTP-Paket-Sender* empfängt das XML-Paket. Das Paket enthält keine Informationen über den Zielserver und die Zugangsdaten, das HTTP-Paket sollte daher diese Einstellungen speichern oder sollte diese On-demand erhalten. Der HTTP-Paket-Sender erstellt einen HTTP-Header mit den Anmeldeinformationen des Panel-Benutzers und sendet das Paket an den Panel-API RPC-Server via API RPC-Protokoll.
3. Der *HTTP-Paket-Empfänger* erhält das resultierende HTTP-Paket von der Serverseite, entfernt den HTTP-Header und gibt die reine XML-Antwort an den XML-Paket-Parser weiter.
4. Der *XML-Paket-Parser* analysiert den Ausführungsstatus in dem XML-Antwortpaket und erstellt eine entsprechende Fehlernachricht, falls die Antwort negativ ist. Es sendet anschließend die extrahierten und verarbeiteten Daten an die GUI oder an das anfragende Modul zurück.

Die Abbildung oben zeigt die HTTP-Operationen als Teil der HTTP-Bibliothek. Diese Operationen können von Grund auf durchgeführt werden, aber es ist einfacher eine einsatzbereite und getestete HTTP-Bibliothek zu verwenden. Die Einteilung in HTTP-Sender und HTTP-Empfänger ist konditional: Diese wurde gemacht, um die HTTP-Operationen *senden* und *empfangen* grafisch voneinander zu unterscheiden.

In diesem Abschnitt:

Erstellen eines Abfrage-Pakets 58
Abfragenachrichten versenden 59
Analysieren der Antwort 61
Fehlerhandling 63
Client Code-Beispiele 65

Erstellen eines Abfrage-Pakets

Beim Erstellen eines API RPC-Abfrage-Pakets erhält die Client-Anwendung von der grafischen Benutzeroberfläche oder von einem Softwaremodul den Befehl, einen bestimmten Vorgang (oder mehrere) auf einem oder mehrere Panel-Objekten auszuführen. Die Anwendung muss den angefragten Befehl in das Format eines XML-Pakets konvertieren, das mit der Panel XML-API kompatibel ist.

Um den “Command-to-XML”-Paket-Konverter, zu implementieren, beachten Sie bitte die Struktur von API RPC-Paketen und wie diese vom Panel-Server verarbeitet werden (auf Seite 53).

Um die Validierung des erstellten Abfrage-Pakets zu Ihrem Konverter hinzuzufügen, verwenden Sie bitte das Set an XML-Schemadateien aus Ihrer Panel-Installation:

- `PRODUCT_ROOT_D/admin/htdocs/schemas/rpc/1.6.2.0` - auf Linux/Unix
  wobei `PRODUCT_ROOT_D` eine Variable darstellt, die das Verzeichnis für die Panel-Installation definiert.
- `%plesk_dir%\admin\htdocs\schemas\rpc\1.6.2.0` - auf Windows
  wobei `%plesk_dir%` eine Variable darstellt, die das Verzeichnis für die Panel-Installation definiert.

Verwenden Sie unbedingt `agent_input.xsd` als Einsprungspunkt für die Validierung.

Abfragenachrichten versenden

Sobald das Panel API RPC-Paket bereit ist, sollte es in den HTTP-Header eingebunden werden und an den genannten Server gesendet werden. Um diese Tasks auszuführen, empfehlen wir die Verwendung einer HTTP/FTP-Client-Bibliothek. In diesem Abschnitt erfahren Sie, wie dieser Vorgang in PHP unter Verwendung des CURL-Engines durchgeführt werden kann. Es handelt sich um eine kostenlose und frei verfügbare, clientseitige URL-Transferbibliothek, die in PHP 4.0.2 und höher unterstützt wird.

Um den HTTP-Header zu erstellen, benötigt der CURL-Engine die folgenden Parameter:

- Die URL des Zielservers in einem String in dem folgenden Format:

```
   define ('HOST', '10.58.97.81');
   define ('PORT', 8443);
   define ('PATH', 'enterprise/control/agent.php');
…
$url = 'https://' . HOST . ':' . PORT . '/' . PATH;
```

- Die Anordnung der Header-Elemente ist wie folgt:

```
$headers = array(
   'HTTP_AUTH_LOGIN: admin',
   'HTTP_AUTH_PASSWD: setup',
'Content-Type: text/xml'
);
```

Dann wird der CURL-Engine initialisiert, so eingerichtet, dass HTTPS verwendet werden kann und anschließend werden alle für den HTTP-Header erforderlichen Parameter eingegeben:

```
// initialize the curl engine
$ch = curl_init();


// set the curl options:

// do not check the name of SSL certificate of the remote server
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);

// do not check up the remote server certificate curl_setopt($ch,
CURLOPT_SSL_VERIFYPEER, false);

// pass in the header elements curl_setopt($ch, CURLOPT_HTTPHEADER,
$headers);

// pass in the url of the target server
```

```
curl_setopt($ch, CURLOPT_URL, $url);
```

Ein weiterer Parameter, der vom CURL-Engine benötigt wird, ist das Panel API RPC-Paket. Nehmen wir an, dass das Paket wie folgt strukturiert ist:

```
$packet = <<<EOP
<?xml version="1.0" encoding="UTF-8"?>
<packet version="1.6.2.0">
<smb>
  <initial_setup>
    <password>P4$$w0rd</password>
    <admin_email>johndoe@example.com</admin_email>
  </initial_setup>
</smb>
</packet>
EOP;
```

Jetzt geben wir an den CURL-Engine die Information weiter, dass wir eine Abfrage von dem Server erwarten, veranlassen die Übermittlung des Pakets und starten die Übertragung:

```
// tell CURL to return the result rather than to load it to the browser
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
// pass in the packet to deliver
curl_setopt($ch, CURLOPT_POSTFIELDS, $packet);

// perform the CURL request and return the result
$retval = curl_exec($ch);
// close the CURL session
curl_close($ch);
```

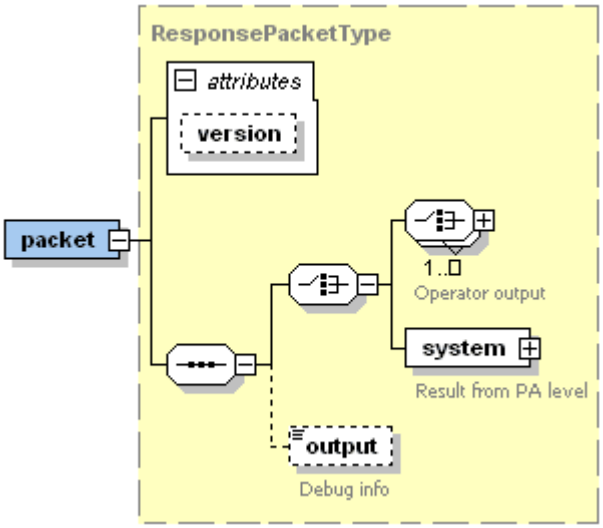
Sobald die Abfrage durchgeführt wurde, erhält CURL das resultierende HTTP-Paket auf der Clientseite, entfernt den HTTP-Header und die Variable `$retval` erhält den reinen XML-Teil des 'Antwort'-Pakets.

Analysieren der Antwort

Sie können eine beliebige HTTP/FTP-Client-Bibliothek für die Übertragung der Abfrage an das Panel verwenden und für den Erhalt der Antwort. In der Regel wird das resultierende HTTP-Paket so strukturiert sein, wie im Abschnitt **Paketstruktur** beschrieben (auf Seite 54). Falls der CURL-Engine verwendet wird, wird das Ergebnis im reinen XML-Format wieder zurückgegeben.

Die Struktur eines reinen XML-Pakets, das von dem Panel API RPC-Server wiedergegeben wird, wird von den XML-Schemata definiert.

Das Element **packet** ist der Root-Knoten eines jeden XML-Antwortpakets, das Attribut **version** gibt die Version des Panel API RPC-Protokolls wieder, die auf dem Server ausgeführt wird.



Ein Antwortpaket enthält entweder die Antwort von bestimmten Operatoren oder den Fehler der Panel-Anwendung, die in dem Systemknoten **system** wiedergegeben wird.

Detaillierte Informationen zu den Bedeutungen der Knoten **system** und **output** finden Sie in dem Abschnitt **Fehlerhandling** (auf Seite 63).

Um die Struktur der Antworten zu analysieren, benötigen Sie das Schema `agent_output.xsd`.Dieses Schema zählt die *Antwortdatentypen* für alle Operatoren auf, die von einer bestimmten Version von API RPC unterstützt werden, und gibt zudem die Ausgabeschemata an, die mit diesen Operatoren übereinstimmen.

Innerhalb eines jeden *[operator]*-Elements eines Antwortpakets ist ein typischer Antwortdatentyp enthalten, das ein **result**-Element oder oder mehrere beinhaltet. Dieses Element wird über *complex type* dargestellt und beinhaltet typischerweise den Typ *resultType*, der in dem Schema `common.xsd` wie folgt definiert wird:

<xs:complexType name=”resultType”>

```
<xs:sequence>
  <xs:element name=”status” type=”result_status”></xs:element>
  <xs:element name=”errcode” type=”unsignedInt”
minOccurs=”0”></xs:element>
```

```
  <xs:element name="errtext" type="xs:string"
minOccurs="0"></xs:element>
</xs:sequence>
</xs:complexType>
```

Das **result**-Element kann zudem weitere optional eingeschachtelte Elemente enthalten, sofern diese in dem complexType definiert wurden.

In den meisten Fällen sieht das Format des Pakets, das an den Panel API RPC-Server zurückgegeben wird, wie folgt aus:

```
<?xml="1.0" encoding="UTF-8" ?>
<packet version="1.6.2.0">
<[operator]>
  <[operation]>
    <result>
          ...
    </result>
[
  <result>
          ...
  </result>
          ...
  <result>
          ...
  </result>
]
  </[operation]>
</[operator]>
</packet>
```

In dem obigen Beispiel steht *[operator]* für den Operator, der in dem Abfrage-Paket verwendet wird und *[operation]* bezieht sich auf die durchgeführte Operation. Das RESULT-Element wird *immer* verwendet, um die erforderlichen Information und den Ausführungsstatus von der durchgeführten Operation zu isolieren. Der Abschnitt RESULT enthält immer das STATUS-Element, das den Ausführungsstatus der durchgeführten Operation enthält.

Hinweis: Der Funktionsbereich einer echten XML-Abfrage kann mehrere Funktionen enthalten, wenn das Eingangspaket für mehrere Funktionen angefordert wurde. Ein Antwortpaket kann mehrere Funktionsbereiche enthalten, wenn das Eingangspaket für Aufgaben mehrerer Panel-Objekte benötigt wird.

Fehlerhandling

Der Server-Teil berichtet über zwei Fehlertypen an die Clientsoftware, dazu zählen die *Überprüfungsfehler* und die *Ausführungsfehler* (falls es dazu kommt, dass die angeforderte Operation ausgeführt wird).

Falls ein Fehler dieser Art auftritt, erhält der von dem Panel verwaltete Server eine Fehlermeldung. Die Fehlerbeschreibung beinhaltet in dem Fall immer die folgenden Parameter:

- *status* mit dem Wert “error”
- *errcode* – Fehlercode, eine eindeutige Nummer zur Identifikation des Fehlers
- *errtext* – Fehlermeldung, eine für den Menschen lesbare Fehlerbeschreibung

Eine Fehlerbeschreibung wird von dem Typ *resultType* in dem `common.xsd`-Schema (in den meisten Fällen) oder von Typen basierend auf diesem Schema definiert (abhängig von dem Operator).

Überprüfungsfehler

Ein eingehendes HTTP-Paket durchläuft eine Reihe von Überprüfungen auf der Serverseite bevor es als gültig bzw. valide anerkant wird und für die Ausführung bereit ist. Hier sind einige dieser Überprüfungen:

- Überprüfung des HTTP-POST-Headers
- Überprüfung, ob die angegebene Version des API RPC-Protokolls unterstützt wird oder nicht
- Überprüfung der Authentifizierungsmethode (Anmeldeinformationen oder geheimer Schlüssel)
- Überprüfung des Berechtigungsstatus des Benutzers ‘remote_access_interface’
- ...
- Überprüfung der Validität des Pakets
- Überprüfung, ob die benötigten Agents in der aktuellen Panel-Version verfügbar sind

Jede dieser Überprüfungen kann unter Umständen fehlschlagen. In dem Fall können keine weiteren Überprüfungen und Befehle ausgeführt werden. Wenn die Serverseite bei einem der vorhergehenden Überprüfungen “hängen” bleibt (z.B. die angefragte Operation konnte bisher noch nicht ausgeführt werden), dann wird ein Antwortpaket mit den folgenden Elementen erstellt:

- **system** enthält eine Fehlerbeschreibung; definiert von dem Typ *resultType* in dem `common.xsd`-Schema
- **output**, ein optionales Element, dass eine Nachricht von stderr (sofern nicht leer) der fehlgeschlagenen Komponente enthält; definiert von dem Typ *garbageOutput* in dem `plesk_common.xsd`-Schema

Beispiel: Die Antwort, die von dem Panel API RPC-Server empfangen wurde, beim Versuch eine nicht-existente Version des API RPC-Protokolls zu verwenden, sieht wie folgt aus:

```
<packet version="1.60.2.0">
  <system>
   <status>error</status>
   <errcode>1005</errcode>
   <errtext>API RPC protocol version not supported.</errtext>
   </system>
   <output>…</output>
</packet>
```

Ausführungsfehler

Wenn alle Überprüfungen erfolgreich durchgeführt werden konnten, werden die ausgewählten Agents versuchen, die angefragten Operationen der Reihe nach auszuführen. Falls eine Operation fehlschlägt, berichtet der entsprechende Agent über den Ausführungsfehler. Alle Berichte werden vom "Agent Engine" gesammelt und dieser erstellt ein Antwortpaket basierend auf den Ausgabedaten der XML-Schemata. Wenn beispielsweise keine Domain erstellt werden konnte, dann erstellt der **Domain**-Operator eine Fehlermeldung mithilfe des `domain_output.xsd`-Schemas:

```
<packet version="1.6.2.0">
<domain>
<add>
  <result>
   <status>error</status>
   <errcode>2300</errcode>
   <errtext>Failed to add domain.</errtext>
  </result>
</add>
</domain>
</packet>
```

Client Code-Beispiele

In diesem Abschnitt finden Sie Beispiele für Client-Anwendungen in PHP, C# und VB.NET, die erläutern, wie das API RPC-Protokoll verwendet wird.

In diesem Abschnitt:

PHP Client-Anwendung 65
C# Client-Anwendung 71
VB.NET Client-Anwendung 79

PHP Client-Anwendung

Die folgenden Code-Beispiele stellen eine einsatzbereite Client-Anwendung dar, die in PHP geschrieben wurde.

Hinweis: Wenn Sie dieses Beispiel verwenden möchten, benötigen Sie PHP 4.0.2 oder höher.
Weitere Informationen finden Sie unter http://php.net/manual/en/ref.dom.php oder http://php.net/manual/en/ref.curl.php.

Kommentare

`class ApiRequestException`	Erweitert die standardmäßige “Exception”-Klasse. Weitere Einzelheiten finden Sie unter http://www.php.net/manual/en/language.exceptions.php.	
`function domainsInfoRequest()`	Verwendet das DOM-Modell zur Erstellung eines XML-Abfrage-Pakets. Gibt das resultierende DOM-Objekt zurück.	
`function curlInit()`	Initialisiert die CURL-Sitzung mit den Optionen wie folgt:	
	CURLOPT_URL	benennt die Ziel-URL
	CURLOPT_RETURNTRANSFER	der Wert *true* bedeutet, dass die resultierende Ausgabe von dem Server in Form eines *Strings* zurückgegeben wird.
	CURLOPT_POST	der Wert *true* bedeutet, dass das Paket per HTTP-POST versendet wird.

	CURLOPT_SSL_VERIFYPEER	der Wert *false* stoppt CURL bei der Verifizierung des Peer-Zertifikats.
	CURLOPT_SSL_VERIFYHOST	der Wert *false* stoppt CURL bei der Verifizierung des Hosts.
	CURLOPT_HTTPHEADER	definiert eine Reihe von einzurichtenden Feldern des HTTP-Headers.
	Gibt den Handler der URL-Sitzung zurück.	
function sendRequest()	Sendet das HTTP-Paket mithilfe von CURL zurück und empfängt das reine XML-Antwortpaket (ohne den HTTP-Header). Schließt die CURL-Sitzung und gibt das resultierende Paket zurück (*Nur-Text* mit XML-Tags). Weitere Informationen zum CURL-Engine erhalten Sie unter http://www.php.net/manual/en/ref.curl.php.	
function parseResponse()	Erhält das Antwortpaket (*Nur-Text*) als einen Parameter. Verarbeitet das Paket mithilfe von SimpleXML und gibt das Objekt *SimpleXMLElement* zurück, indem das Paket in der Baumstruktur dargestellt ist. Um mehr über die SimpleXML-Erweiterung von PHP zu erfahren, besuchen Sie bitte folgende Website: http://www.php.net/manual/en/ref.simplexml.php.	
function checkResponse()	Erhält das Antwortpaket *SimpleXMLEelement* und überprüft den **result**-Knoten. Falls in diesem Knoten ‘error’ steht, dann schreibt die Funktion eine Ausnahme mit dem String in den **errtext** -Knoten.	

Die main()-Funktion führt diese Funktionen in der Reihenfolge auf, in der sie berücksichtigt werden. Zuerst wird domainInfoRequest() aufgerufen, um ein Paket wie folgt zu erstellen:

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<packet version=”1.6.2.0”>
<domain>
<get>
    <filter/>
    <dataset>
        <limits/>
        <prefs/>
        <user/>
```

```
        <hosting/>
        <stat/>
        <gen_info/>
  </dataset>
</get>
</domain>
</packet>
```

Dann wird curlInit() aufgerufen, um eine CURL-Sitzung mit Optionen zu initialisieren. Die Funktion sendRequest versendet ein Paket mit dem HTTP-Header wie folgt:

```
POST /enterprise/control/agent.php HTTP/1.1
Host: 10.58.32.100:8443
HTTP_AUTH_LOGIN: Login
HTTP_AUTH_PASSWD: qwedsa
HTTP_PRETTY_PRINT: TRUE
Content-Length: 294
Content-Type: text/xml
```

Das XML-Antwortpaket, das von dem vom Panel verwalteten Server empfangen wird, sieht wie folgt aus:

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<packet version=”1.6.2.0”>
<domain>
<get>
  <result>
    <status>ok</status>
    <id>1234</id>
    <value>
      <gen_info>
    <cr_date>1154513574</cr_date>
    <name>example.com</name>
    <display_name>example</display_name>
    <status>64</status>
    <real_size>0</real_size>
    <client_id>46</client_id>
    <dns_ip_address>192.0.2.33</dns_ip_address>
```

```
    <htype>none</htype>
     </gen_info>
     </value>
   </result>
</get>
</domain>
</packet>
```

Sobald das Antwortpaket empfangen und mit der parseResponse()-Funktion geparst wurde, gibt der foreach-Iterator die Inhalte des **result** -Knotens wieder.

Code-Beispiel

```
<?php

/**
 ▪ Reports error during API RPC request
 */
class ApiRequestException extends Exception {}

/**
 ▪ Returns DOM object representing request for information about all available domains
 ▪ @return DOMDocument
 */

function domainsInfoRequest()
{
  $xmldoc = new DomDocument('1.0', 'UTF-8');
  $xmldoc->formatOutput = true;
  // <packet>
  $packet = $xmldoc->createElement('packet');
  $packet->setAttribute('version', '1.6.2.0');
  $xmldoc->appendChild($packet);
  // <packet/domain>
  $domain = $xmldoc->createElement('domain');
  $packet->appendChild($domain);
    // <packet/domain/get>
    $get = $xmldoc->createElement('get');
    $domain->appendChild($get);
```

```
  // <packet/domain/get/filter>
  $filter = $xmldoc->createElement('filter');
  $get->appendChild($filter);
  // <packet/domain/get/dataset>
  $dataset = $xmldoc->createElement('dataset');
  $get->appendChild($dataset);
  // dataset elements
  $dataset->appendChild($xmldoc->createElement('hosting'));
  $dataset->appendChild($xmldoc->createElement('gen_info'));
  return $xmldoc;
}

/**
```

- Prepares CURL to perform the Panel API request
- @return resource

```
 */
```

function curlInit($host, $login, $password)

```
{
  $curl = curl_init();
  curl_setopt($curl, CURLOPT_URL,
"https://{$host}:8443/enterprise/control/agent.php");
  curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($curl, CURLOPT_POST,           true);
  curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, false);
  curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, false);
  curl_setopt($curl, CURLOPT_HTTPHEADER,
    array("HTTP_AUTH_LOGIN: {$login}",
   "HTTP_AUTH_PASSWD: {$password}",
   "HTTP_PRETTY_PRINT: TRUE",
   "Content-Type: text/xml")
      );

  return $curl;
}

/**
```

- Performs a Panel API request, returns raw API response text

```
 *
```

- @return string
- @throws ApiRequestException

```
 */
```

function sendRequest($curl, $packet)

```
{
  curl_setopt($curl, CURLOPT_POSTFIELDS, $packet);
    $result = curl_exec($curl);
    if (curl_errno($curl)) {
        $errmsg  = curl_error($curl);
        $errcode = curl_errno($curl);
        curl_close($curl);
     throw new ApiRequestException($errmsg, $errcode);
       }
    curl_close($curl);
    return $result;
}

/**
 *  Looks if API responded with correct data
 *
 *  @return SimpleXMLElement
 *  @throws ApiRequestException
 */
function parseResponse($response_string)
{
   $xml = new SimpleXMLElement($response_string);
   if (!is_a($xml, 'SimpleXMLElement'))
  throw new ApiRequestException("Cannot parse server response:
{$response_string}");
  return $xml;
}

/**
 *  Check data in API response
 *  @return void
 *  @throws ApiRequestException
 */
function checkResponse(SimpleXMLElement $response)
{
  $resultNode = $response->domain->get->result;
  // check if request was successful

   if ('error' == (string)$resultNode->status)
  throw new ApiRequestException("The Panel API returned an error: " .
(string)$resultNode->result->errtext);
}

//
```

```
// int main()
//
$host = '10.58.32.100';
$login = 'admin';
$password = 'qwedsa';

$curl = curlInit($host, $login, $password);

try {
   $response = sendRequest($curl, domainsInfoRequest()->saveXML());
    $responseXml = parseResponse($response);
    checkResponse($responseXml);

} catch (ApiRequestException $e) {
    echo $e;
    die();
}

// Explore the result
foreach ($responseXml->xpath('/packet/domain/get/result') as
$resultNode) {
      echo "Domain id: " . (string)$resultNode->id . " ";
  echo (string)$resultNode->data->gen_info->name . " (" .
(string)$resultNode->data->gen_info->dns_ip_address . ")\n";
}

?>
```

C# Client-Anwendung

Die folgenden Code-Beispiele stellen eine einsatzbereite Client-Anwendung dar, die in C# geschrieben wurde.

Kommentare

Die *Request*-Klasse versendet ein Abfrage-Paket an den vom Panel verwalteten Server per HTTP und empfängt das Antwortpaket. Es enthält die folgenden Elemente:

`public string Hostname`	Beinhaltet den Host-Namen (IP-Adresse) des vom Panel verwalteten Servers, an den das Abfrage-Paket versendet wird. Standardmäßig handelt es sich dabei um "localhost".
`public string Login`	Beinhaltet den Login des Panel-Administrators. Standardmäßig handelt es sich dabei um "admin".

`public string Password`	Beinhaltet das Passwort des Panel-Administrators. Standardmäßig handelt es sich dabei um “setup”.
`public string Protocol`	Beinhaltet die Version des API RPC-Protokolls, das für die Kommunikation mit dem Panel verwendet wird.
`public ValidationEventHandler XmlSchemaValidation`	Der verwendete Handler für den Empfang von Schema-Validierungsfehlern.
`public string AgentEntryPoint`	Beinhaltet die URL des Panel Agents, der das Abfrage-Paket auf der Server-Seite verarbeiten wird.
`public string InputValidationSchema`	Beinhaltet die URL des Validierungsschemas, die in dem Abfrage-Paket übernommen wird, bevor es an die Server-Seite versendet wird.
`public string OutputValidationSchema`	Beinhaltet die URL des Validierungsschemas, die in dem Antwortpaket übernommen wird, bevor es auf der Client-Seite empfangen wird.
`public XmlDocument Send (XmlDocument)`	Leitet das Abfrage-Paket (in Form des XmlDocument-Objekts) an die Eingabe-Parameter weiter. Versendet eine Anfrage und erhält die Antwort in Form des XmlDocument-Objekts.
`public XmlDocument Send (Stream)`	Leitet das Abfrage-Paket (Stream) an die Eingabe-Parameter weiter. Validiert das Abfrage-Paket mithilfe des Validierungsschemas `agent_input.xsd`. Ruft die Elementfunktion `Send(XmlDocument)` auf.
`public XmlDocument Send (string)`	Leitet die URI des Abfrage-Pakets (XML-Datei) an die Eingabe-Parameter weiter. Validiert das Abfrage-Paket mithilfe des Validierungsschemas `agent_input.xsd`. Ruft die Elementfunktion `Send(XmlDocument)` auf.
`private HttpWebRequest SendRequest (string)`	Erstellt eine HTTP-Abfrage: Überträgt den HTTP-Header und das serialisierte XML-Paket auf das Objekt des Typs HttpWebRequest. Schickt dieses Objekt zurück.
`private XmlDocument ParseAndValidate (TextReader, string)`	Leitet die URI des `agent_input.xsd` Schemas und einen String Reader (mit der URI des zu validierenden XML-Pakets) an die Eingabe-Parameter weiter. Validiert das Paket und gibt es in der Baumstruktur wieder (ein XmlDocument-Objekt).
`private XmlDocument GetResponse (HttpWebRequest)`	Leitet das Abfrage-Paket (HttpWebRequest-Objekt) an die Eingabe-Parameter weiter. Versendet das Paket per HTTP, empfängt das Antwortpaket vom Server, validiert es mithilfe des `agent_output.xsd`-Schemas und gibt das XML-Paket in der Baumstruktur wieder (ein XmlDocument-Objekt).

Die **Program**-Klasse implementiert die ‘client’-Konsolenanwendung.

`static void Main (string[])`	Ein Einsprungspunkt zu der *PanelApiRpcClient* -Anwendung. Leitet eine Reihe an Argumenten (Strings) an die Eingabe-Parameter weiter. Die Argumente sind: [0] - der Host-Name (IP-Adresse) des vom Panel verwalteten Servers, [1] - der Login des Panel-Administrators, [2] - das Passwort des Panel-Administrators, [3] - das verwendete API RPC-Protokoll, [4] - einem Pfad zu der XML-Datei mit einem XML-Abfrage-Paket. Die Funktion erstellt ein Abfrage-Paket des Typs Request (siehe oben), validiert dieses, versendet eine Anfrage an das Panel und liefert das resultierende XML-Paket.
`private static bool RemoteCertificateValidation (object, X509Certificate, X509Chain, SslPolicyErrors)`	Die Funktion verifiziert das Remote-SSL-Zertifikat, um den Server zu authentifizieren. Sofern der Server nicht authentifiziert werden konnte, kommt *false* zurück.
`private static void XmlSchemaValidation (object, ValidationEventArgs)`	Ereignishandler bei Validierungsfehlern
`static void PrintResult(XmlDocument)`	Die Funktion gibt das Antwortpaket (XmlDocument-Objekt) an die Konsole aus.

Die Client-Anwendung kann von der Befehlszeile wie folgt gestartet werden:

```
PanelApiRpcClient 192.0.2.168 admin_login admin_passwd 1.6.2.0
"c:\requests\AddNewDomain.xml"
```

Das Abfrage-Paket wird in der `AddNewDomain.xml`-Datei verarbeitet:

```
<?xml version="1.0" encoding="UTF-8"?>
<packet version="1.6.2.0">
<domain>
  <add>
    <gen_setup>
    <name>example.com</name>
    <ip_address>192.0.2.48</ip_address>
    </gen_setup>
    <hosting>
      <vrt_hst>
    <property>
        <name>ftp_login</name>
        <value>fpt16se4fdf0</value>
        </property>
        <property>
          <name>ftp_password</name>
          <value>qweqwe</value>
        </property>
        <ip_address>192.0.2.48</ip_address>
    </vrt_hst>
    </hosting>
  </add>
  </domain>
</packet>
```

Das Antwortpaket, das vom Server empfangen wird, kann wie folgt aussehen:

```
<?xml version="1.0"?>
<packet version="1.6.2.0">
  <domain>
    <add>
    <result>
```

```
    <status>ok</status>
    <id>6</id>
    <guid>5c0e3881-22a2-4401-bcc0-881d691bfdef</guid>
     </result>
     </add>
  </domain>
</packet>
```

Dieses Paket berichtet über das Ergebnis der **add**-Operation und der ID und GUID von der gerade erstellten Domain.

Code-Beispiel

```
using System;
using System.Net;
using System.Text;
using System.IO;
using System.Xml;
using System.Xml.Schema;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace PanelApiRpcClient
{
  public class Request
      {
    // Public interface
            //
            public string Hostname  = "localhost";     // Panel Hostname
            public string Login     = "admin_login";         //
Administrator's Login
            public string Password  = "admin_passwd";        //
Administrator's Password
            public string Protocol  = "1.6.2.0";       // API RPC Version
Protocol.

  // Handler for receiving information about document type definition
(DTD),
  // XML-Data Reduced (XDR) schema, and XML Schema definition language
(XSD)
   // schema validation errors.
  public ValidationEventHandler XmlSchemaValidation = null;
  public Request()
```

```
            {
            }

  public string AgentEntryPoint        { get { return "https://" +
Hostname + ":8443/enterprise/control/agent.php"; } }
  public string InputValidationSchema  { get { return "https://" +
Hostname + ":8443/schemas/rpc/" + Protocol + "/agent_input.xsd"; } }
  public string OutputValidationSchema { get { return "https://" +
Hostname + ":8443/schemas/rpc/" + Protocol + "/agent_output.xsd"; } }
    public XmlDocument Send(XmlDocument packet)
            {
    HttpWebRequest request = SendRequest(packet.OuterXml);
   XmlDocument result = GetResponse(request);
   return result;
            }
    public XmlDocument Send(Stream packet)
            {
    using (TextReader reader = new StreamReader(packet))
                {
  return Send(ParseAndValidate(reader, InputValidationSchema));
                }
            }
    public XmlDocument Send(string packetUri)
            {
    using (TextReader reader = new StreamReader(packetUri))
                {
  return Send(ParseAndValidate(reader, InputValidationSchema));
                }
            }

    // Private interface
            //

    // Sending a request message
            //
    private HttpWebRequest SendRequest(string message)
            {
  HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(AgentEntryPoint);
   request.Method = "POST"; request.Headers.Add("HTTP_AUTH_LOGIN",
   Login); request.Headers.Add("HTTP_AUTH_PASSWD", Password);
   request.ContentType = "text/xml"; request.ContentLength =
   message.Length;
   ASCIIEncoding encoding = new ASCIIEncoding(); byte[] buffer =
   encoding.GetBytes(message);
   using (Stream stream = request.GetRequestStream())
```

```
                {
    stream.Write(buffer, 0, message.Length);
                }

    return request;
            }

    // Parsing and validating packet
            //
  private XmlDocument ParseAndValidate(TextReader xml, string
schemaUri)
            {
   XmlSchemaSet schemas = new XmlSchemaSet();
   schemas.Add(null, schemaUri);
   XmlReaderSettings settings = new XmlReaderSettings(); if
   (XmlSchemaValidation != null)
  settings.ValidationEventHandler += new
ValidationEventHandler(XmlSchemaValidation);
    settings.ValidationType = ValidationType.Schema;
  settings.ValidationFlags |=
XmlSchemaValidationFlags.ProcessSchemaLocation;
    settings.Schemas = schemas;
    XmlDocument document = new XmlDocument();
  using (XmlReader reader = XmlTextReader.Create(xml, settings))
                {
    document.Load(reader);
                }

    return document;
            }
    private XmlDocument GetResponse(HttpWebRequest request)
            {
  using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
   using (Stream stream = response.GetResponseStream())
   using (TextReader reader = new StreamReader(stream))
                {
  return ParseAndValidate(reader, OutputValidationSchema);
                }
            }
      }

  class Program
      {
    static void Main(string[] args)
```

```
            {
    if (args.Length < 5)
                {
  Console.WriteLine("Usage: PanelApiRpcClient <Hostname> <Login>
<Password> <Protocol> <Request>");
    Console.WriteLine("  ");
    Console.WriteLine("  Hostname  - Panel host name");
  Console.WriteLine("  Login     - Administrator's login");
  Console.WriteLine("  Password  - Administrator's password");
  Console.WriteLine("  Protocol  - API RPC protocol version");
  Console.WriteLine("  Request   - Request file path (*.xml)");
    return;
                }

   // Verifies the remote Secure Sockets Layer (SSL) certificate
   // used for authentication.
   ServicePointManager.ServerCertificateValidationCallback =
  new
RemoteCertificateValidationCallback(RemoteCertificateValidation);
   Request request = new Request(); request.XmlSchemaValidation =
   XmlSchemaValidation;
                request.Hostname  = args[0]; // "10.49.8.120";
                request.Login     = args[1]; // "admin";
                request.Password  = args[2]; // "setup";
                request.Protocol  = args[3]; // "1.6.2.0";

                string packet     = args[4]; // "request.xml";

    try
                {
    XmlDocument result = request.Send(packet);
    PrintResult(result);
                }
    catch (Exception e)
                {
    Console.WriteLine("Request error: {0}", e.Message);
                }
            }

  // The following method is invoked by the
RemoteCertificateValidationDelegate.
    private static bool RemoteCertificateValidation(object sender,
  X509Certificate certificate, X509Chain chain, SslPolicyErrors
sslPolicyErrors)
            {
```

```
  if (sslPolicyErrors !=
SslPolicyErrors.RemoteCertificateNotAvailable)
    return true;
  Console.WriteLine("Certificate error: {0}", sslPolicyErrors);
  // Do not allow this client to communicate with unauthenticated servers.
    return false;
            }
            //
  private static void XmlSchemaValidation(object sender,
ValidationEventArgs e)
            {
    Console.WriteLine("Validation error: {0}", e.Message);
            }

    static void PrintResult(XmlDocument document)
            {
   XmlTextWriter writer = new XmlTextWriter(Console.Out);
   writer.Formatting = Formatting.Indented;
   document.WriteTo(writer);
    writer.Flush();
    Console.WriteLine();
            }
      }
}
```

VB.NET Client-Anwendung

Die folgenden Code-Beispiele stellen eine einsatzbereite Client-Anwendung dar, die in VB.NET geschrieben wurde.

Kommentare

Die *Request*-Klasse versendet ein Abfrage-Paket an den vom Panel verwalteten Server per HTTP und empfängt das Antwortpaket. Es enthält die folgenden Elemente:

`public string Hostname`	Beinhaltet den Host-Namen (IP-Adresse) des vom Panel verwalteten Servers, an den das Abfrage-Paket versendet wird. Standardmäßig handelt es sich dabei um “localhost”.
`public string Login`	Beinhaltet den Login des Panel-Administrators. Standardmäßig handelt es sich dabei um “admin”.
`public string Password`	Beinhaltet das Passwort des Panel-Administrators. Standardmäßig handelt es sich dabei um “setup”.

`public string Protocol`	Beinhaltet die Version des API RPC-Protokolls, das für die Kommunikation mit dem Panel verwendet wird.
`public XmlSchemaValidation`	Der verwendete Handler für den Empfang von Schema-Validierungsfehler.
`Public ReadOnly Property AgentEntryPoint`	Beinhaltet die URL des Panel Agents, der das Abfrage-Paket auf der Server-Seite verarbeiten wird.
`Public ReadOnly Property InputValidationSchema`	Beinhaltet die URL des Validierungsschemas, die in dem Abfrage-Paket übernommen wird, bevor es an die Server-Seite versendet wird.
`Public ReadOnly Property OutputValidationSchema`	Beinhaltet die URL des Validierungsschemas, die in dem Antwortpaket übernommen wird, bevor es auf der Client-Seite empfangen wird.
`Public Function Send(ByVal packet As XmlDocument)`	Leitet das Abfrage-Paket (in Form des XmlDocument-Objekts) an die Eingabe-Parameter weiter. Versendet eine Anfrage und erhält die Antwort in Form des XmlDocument-Objekts.
`Public Function Send(ByVal packet As Stream)`	Leitet das Abfrage-Paket (Stream) an die Eingabe-Parameter weiter. Validiert das Abfrage-Paket mithilfe des Validierungsschemas. Ruft die Elementfunktion `Send(XmlDocument)` auf.
`Public Function Send(ByVal packetUri As String)`	Leitet die URI des Abfrage-Pakets (XML-Datei) an die Eingabe-Parameter weiter. Validiert das Abfrage-Paket mithilfe des Validierungsschemas. Ruft die Elementfunktion `Send(XmlDocument)` auf.
`Private Function SendRequest(ByVal message As String)`	Formt eine HTTP-Abfrage: Überträgt den HTTP-Header und das serialisierte XML-Paket auf das Objekt des Typs HttpWebRequest. Schickt dieses Objekt zurück.
`Private Function ParseAndValidate(ByVal xml As TextReader, ByVal schemaUri As String)`	Leitet die URI des Validierungsschemas und einen String Reader (mit der URI des zu validierenden XML-Pakets) an die Eingabe-Parameter weiter. Validiert das Paket und gibt es in der Baumstruktur wieder (ein XmlDocument-Objekt).
`Private Function GetResponse(ByVal request As HttpWebRequest)`	Leitet das Abfrage-Paket (HttpWebRequest-Objekt) an die Eingabe-Parameter weiter. Versendet das Paket per HTTP, empfängt das Antwortpaket vom Server, validiert es mithilfe des Validierungsschemas und liefert das XML-Paket in der Baumstruktur (ein XmlDocument-Objekt).

Die **Program**-Klasse implementiert die ‘client’-Konsolenanwendung.

`Shared Sub Main(ByVal args As String())`	Ein Einsprungspunkt zu der PanelApiRpcClient-Anwendung. Leitet eine Reihe an Argumenten (Strings) an die Eingabe-Parameter weiter. Die Argumente sind: [0] - der Host-Name (IP-Adresse) des vom Panel verwalteten Servers, [1] - das Login des Panel-Administrators, [2] - das Passwort des Panel-Administrators, [3] - das verwendete API RPC-Protokoll, [4] - einem Pfad zu der XML-Datei mit einem XML-Abfrage-Paket. Die Funktion erstellt ein Abfrage-Paket des Typs Request (siehe oben), validiert dieses, versendet eine Anfrage an das Panel und gibt das resultierende XML-Paket wieder.
`Private Shared Function RemoteCertificateValidation(ByVal sender As Object, ByVal certificate As X509Certificate, ByVal chain As X509Chain, ByVal sslPolicyErrors As SslPolicyErrors)`	Die Funktion verifiziert das Remote-SSL-Zertifikat, um den Server zu authentifizieren. Sofern der Server nicht authentifiziert werden konnte, kommt *false* zurück.
`Private Shared Sub XmlSchemaValidation(ByVal sender As Object, ByVal e As ValidationEventArgs)`	Ereignishandler bei Validierungsfehlern
`Private Shared Sub PrintResult(ByVal document As XmlDocument)`	Die Funktion gibt das Antwortpaket (XmlDocument-Objekt) an die Konsole aus.

Die Client-Anwendung kann von der Befehlszeile wie folgt gestartet werden:

```
PanelApiRpcClient 192.0.2.168 admin_login admin_passwd 1.6.2.0
“c:\requests\request.xml”
```

Das Abfrage-Paket wird in der `request.xml`-Datei verarbeitet:

<?xml version=”1.0” encoding=”UTF-8”?>

```
<packet version=”1.6.2.0”>
<domain>
  <add>
    <gen_setup>
   <name>example.com</name>
   <ip_address>192.0.2.48</ip_address>
   </gen_setup>
   <hosting>
      <vrt_hst>
    <property>
         <name>ftp_login</name>
         <value>fp16se4fdf0</value>
         </property>
         <property>
            <name>ftp_password</name>
            <value>qweqwe</value>
         </property>
         <ip_address>192.0.2.48</ip_address>
    </vrt_hst>
    </hosting>
  </add>
  </domain>
</packet>
```

Das Antwortpaket, das vom Server empfangen wird, kann wie folgt aussehen:

```
<?xml version=”1.0”?>
<packet version=”1.6.2.0”>
  <domain>
    <add>
    <result>
   <status>ok</status>
   <id>6</id>
   <guid>5c0e3881-22a2-4401-bcc0-881d691bfdef</guid>
    </result>
    </add>
  </domain>
</packet>
```

Dieses Paket berichtet über das Ergebnis der **add**-Operation und der ID und GUID von der gerade erstellten Domain.

Code-Beispiel

Imports System

Imports System.Net

Imports System.Text

Imports System.IO

Imports System.Xml

Imports System.Xml.Schema

Imports System.Security.Cryptography.X509Certificates

Imports System.Net.Security

Namespace ParallelsPanelApiRpcClient

```
  Public Class Request
```

‘ Public interface

```
            Public Hostname As String = “localhost”   ‘The Panel Host name
            Public Login As String = “admin_login”
      ‘Administrator’s Login
            Public Password As String = “admin_setup”
      ‘Administrator’s Password
            Public Protocol As String = “1.6.2.0”     ‘API RPC Version
Protocol
```

```
  ‘ Handler for receiving information about document type definition
(DTD),
  ‘ XML-Data Reduced (XDR) schema, and XML Schema definition language
(XSD)
    ‘ schema validation errors.
    Public XmlSchemaValidation As ValidationEventHandler
    Public ReadOnly Property AgentEntryPoint As String
```

Get

```
  Return (“https://” & Me.Hostname &
“:8443/enterprise/control/agent.php”)
    End Get
    End Property
    Public ReadOnly Property InputValidationSchema As String
```

Get

```
  Return String.Concat(New String() { "https://", Me.Hostname,
":8443/schemas/rpc/", Me.Protocol, "/agent_input.xsd" })
    End Get
    End Property
    Public ReadOnly Property OutputValidationSchema As String
```

Get

```
  Return String.Concat(New String() { "https://", Me.Hostname,
":8443/schemas/rpc/", Me.Protocol, "/agent_output.xsd" })
    End Get
    End Property
    Public Function Send(ByVal packet As XmlDocument) As XmlDocument
  Dim request As HttpWebRequest = Me.SendRequest(packet.OuterXml)
    Return Me.GetResponse(request)
    End Function
    Public Function Send(ByVal packet As Stream) As XmlDocument
    Using reader As TextReader = New StreamReader(packet)
  Return Me.Send(Me.ParseAndValidate(reader,
Me.InputValidationSchema))
    End Using
    End Function
    Public Function Send(ByVal packetUri As String) As XmlDocument
    Using reader As TextReader = New StreamReader(packetUri)
  Return Me.Send(Me.ParseAndValidate(reader,
Me.InputValidationSchema))
```

End Using

```
    End Function
    ' Private interface
            '
```

' Sending a request message

```
            '
  Private Function SendRequest(ByVal message As String) As
HttpWebRequest
```

```
  Dim request As HttpWebRequest =
DirectCast(WebRequest.Create(Me.AgentEntryPoint), HttpWebRequest)
   request.Method = "POST"
   request.Headers.Add("HTTP_AUTH_LOGIN", Me.Login)
   request.Headers.Add("HTTP_AUTH_PASSWD", Me.Password)
   request.ContentType = "text/xml"
   request.ContentLength = message.Length
   Dim bytes As Byte() = New ASCIIEncoding().GetBytes(message)
   Using stream As Stream = request.GetRequestStream
   stream.Write(bytes, 0, message.Length)
    End Using
    Return request
     End Function
```

‘ Parsing and validating packet

```
            `
  Private Function ParseAndValidate(ByVal xml As TextReader, ByVal
schemaUri As String) As XmlDocument
   Dim schemas As New XmlSchemaSet
   schemas.Add(Nothing, schemaUri)
   Dim settings As New XmlReaderSettings
   If (Not Me.XmlSchemaValidation Is Nothing) Then
  AddHandler settings.ValidationEventHandler, New
ValidationEventHandler(AddressOf Me.XmlSchemaValidation.Invoke)
     End If
     settings.ValidationType = ValidationType.Schema
  settings.ValidationFlags = (settings.ValidationFlags Or
XmlSchemaValidationFlags.ProcessSchemaLocation)
   settings.Schemas = schemas
   Dim document As New XmlDocument
   Using reader As XmlReader = XmlReader.Create(xml, settings)
   document.Load(reader)
    End Using
    Return document
     End Function
  Private Function GetResponse(ByVal request As HttpWebRequest) As
XmlDocument
  Using response As HttpWebResponse = DirectCast(request.GetResponse,
HttpWebResponse)
```

```
    Using stream As Stream = response.GetResponseStream
  Using reader As TextReader = New StreamReader(stream)
  return Me.ParseAndValidate(reader, Me.OutputValidationSchema)
    End Using
    End Using
    End Using
    End Function
```

End Class

Friend Class Program

```
    Shared Sub Main(ByVal args As String())
```

If (args.Length < 5) Then

```
  Console.WriteLine("Usage: PanelApiRpcClient <Hostname> <Login>
<Password> <Protocol> <Request>")
    Console.WriteLine(" ")
    Console.WriteLine(" Host name - The Panel's host name")
    Console.WriteLine(" Login - Administrator's login")
  Console.WriteLine(" Password - Administrator's password")
  Console.WriteLine(" Protocol - API RPC protocol version")
  Console.WriteLine(" Request - Request file path (*.xml)")
```

Else

```
  ' Verifies the remote Secure Sockets Layer (SSL) certificate
    ' used for authentication.
  ServicePointManager.ServerCertificateValidationCallback = New
RemoteCertificateValidationCallback(AddressOf
Program.RemoteCertificateValidation)
```

Dim request As New Request

```
  request.XmlSchemaValidation = New ValidationEventHandler(AddressOf
Program.XmlSchemaValidation)
                        request.Hostname = args(0)                ' 
"10.49.8.120";
                        request.Login = args(1)                   ' "admin";
```

```
                        request.Password = args(2)                ' "setup";
                        request.Protocol = args(3)                '
"1.6.2.0";
                        Dim packetUri As String = args(4)    ' "request.xml";
                        Try
                              Program.PrintResult(request.Send(packetUri))
                        Catch exception As Exception
  Console.WriteLine("Request error: {0}", exception.Message)
    End Try
  End If
  End Sub
  ' The following method is invoked by the
RemoteCertificateValidationDelegate.
  Private Shared Function RemoteCertificateValidation(ByVal sender As
Object, ByVal certificate As X509Certificate, ByVal chain As X509Chain,
ByVal sslPolicyErrors As SslPolicyErrors) As Boolean
  If (sslPolicyErrors <>
SslPolicyErrors.RemoteCertificateNotAvailable) Then
    Return True
    End If
  Console.WriteLine("Certificate error: {0}", sslPolicyErrors)
  ' Do not allow this client to communicate with unauthenticated servers.
```

Return False

```
  End Function
          '
  Private Shared Sub XmlSchemaValidation(ByVal sender As Object, ByVal
e As ValidationEventArgs)
    Console.WriteLine("Validation error: {0}", e.Message)
  End Sub
  Private Shared Sub PrintResult(ByVal document As XmlDocument)
    Dim w As New XmlTextWriter(Console.Out)
    w.Formatting = Formatting.Indented
    document.WriteTo(w)
    w.Flush
    Console.WriteLine
  End Sub
```

End Class

End Namespace

Einrichten des Panels

Dieser Abschnitt erklärt ausführlich, wie jeder Setup-Vorgang des Panels über die Remot-API durchgeführt wird.

In diesem Abschnitt:

Initialisierung des Panels 89
Installation des Lizenzschlüssels 94
DNS-Konfiguration 100
Installation von SSL-Zertifikaten 128
Erstellen von Domains 135
Erstellen von Subdomains 141
Verfügbare APS-Kataloge definieren 146
Installieren von Anwendungen 150

Initialisierung des Panels

Um das Panel per API RPC zu initialisieren, führen Sie das XML-Abfrage-Paket mit dem **smb/initial_setup**-Kontrollknoten aus:

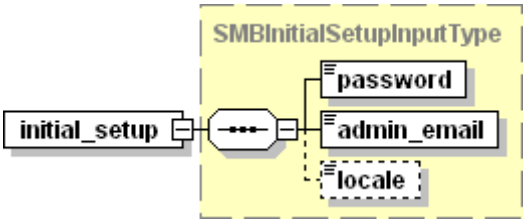
```
<packet version="1.6.2.0">
<smb>
  <initial_setup>
    ...
  </initial_setup>
</smb>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets 90
Beispiele für Abfragen 91
Struktur des Antwortpakets 92
Antwort-Beispiele 93

Struktur des Abfrage-Pakets

Der **initial_setup**-Knoten ist wie folgt strukturiert:



- **password**, *erforderlich*
  Gibt das neue Passwort des Panel-Administrators an. Datentyp: *serverPassword* (`plesk_server.xsd`). String mit 5 bis zu 14 Zeichen.
- **admin_email**, *erforderlich*
  Gibt die E-Mail-Adresse des Panel-Administrators an. Datentyp: *emailType* (`common.xsd`). String mit 3 bis zu 255 Zeichen.
- **locale**, *optional*
  Gibt das Panel-Gebietsschema an - den Ländercode für die Sprache und die Kultur, in dem das Panel-Oberfläche verwendet wird. Datentyp: *string*.

Beispiele für Abfragen

Ein Abfrage-Paket zur Initialisierung des Panels sieht wie folgt aus:

<packet version=”1.6.2.0”>

<smb>

```
  <initial_setup>
    <password>P4$$w0rd</password>
    <admin_email>johndoe@example.com</admin_email>
  </initial_setup>
```

</smb>

</packet>

Ein Abfrage-Paket, über welches das Panel initialisiert wird und das Deutsch (Deutschland) als Standardsprache konfiguriert, sieht wie folgt aus:

<packet version=”1.6.2.0”>

<smb>

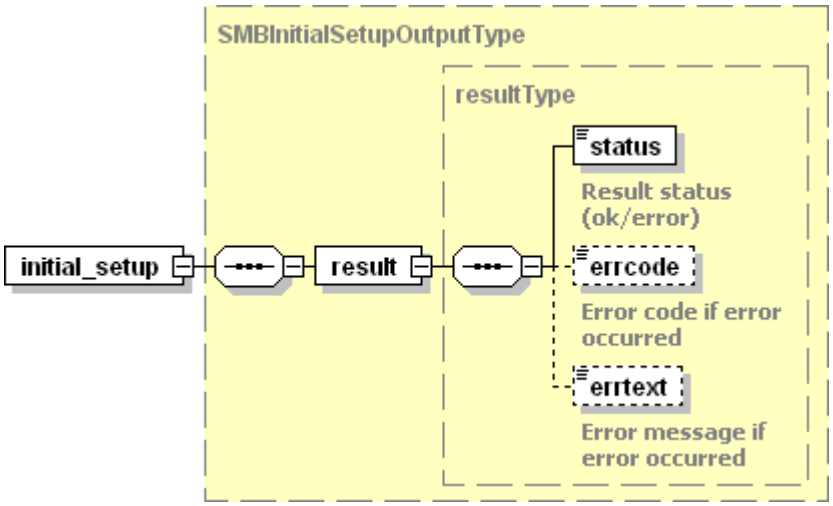
```
  <initial_setup>
    <password>P4$$w0rd</password>
    <admin_email>johndoe@example.com</admin_email>
    <locale>de-DE</locale>
  </initial_setup>
```

</smb>

</packet>

Struktur des Antwortpakets

Der **initial_setup** -Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.

Operations-spezifische Fehler

- 1003, Initialisierungsfehler des Agent Engines.
  Das Panel ist bereits initialisiert.
- 1019, ungültiger Werte.
  Das angegebene Gebietsschema ist nicht verfügbar.

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<smb>
    <initial_setup>
    <result>
     <status>ok</status>
    </result>
    </initial_setup>
</smb>
</packet>
```

Die folgende negative Antwort wird vom Panel empfangen, falls es bereits initialisiert wurde.

```
<packet version=”1.6.2.0”>
<smb>
    <initial_setup>
    <result>
        <status>error</status>
        <errcode>1003</errcode>
     <errtext>Initial setup already completed</errtext>
    </result>
    </initial_setup>
</smb>
</packet>
```

Die folgende negative Antwort wird vom Panel empfangen, falls das angegebene Gebietsschema nicht verfügbar ist.

```
<packet version=”1.6.2.0”>
<smb>
    <initial_setup>
    <result>
        <status>error</status>
        <errcode>1019</errcode>
     <errtext>Locale en-US2 is not installed on server</errtext>
    </result>
```

```
    </initial_setup>
</smb>
</packet>
```

Installation des Lizenzschlüssels

Um einen Lizenzschlüssel per API RPC zu installieren, erstellen Sie ein XML-Abfrage-Paket mit dem **server/lic_install**-Kontrollknoten:

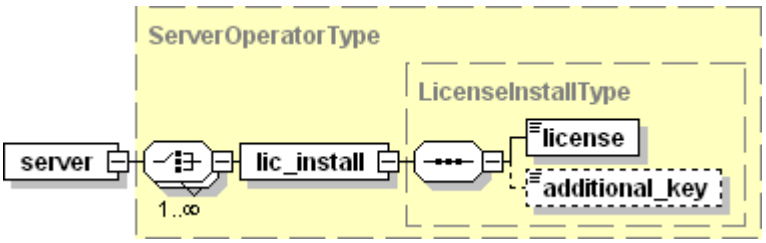
```
<packet version="1.6.2.0">
<server>
  <lic_install>
    ...
  </lic_install>
</server>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets 94
Beispiele für Abfragen 95
Struktur des Antwortpakets 98
Antwort-Beispiele 99

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket zur Installation eines Lizenzschlüssels beinhaltet den **server/lic_install**-Kontrollknoten und ist wie folgt strukturiert:



- **license**, *erforderlich*
  Beinhaltet die Lizenzschlüssel-Daten. Datentyp: *base64*.
- **additional_key**, *optional*
  Gibt an, ob der zu installierende Lizenzschlüssel ein zusätzlicher Schlüssel ist. Datentyp: *none*.

Beispiele für Abfragen

Dieses Paket installiert einen Standard-(Haupt-) Lizenzschlüssel für Parallels Small Business Panel für Linux/Unix, mit dem das Produkt verteilt wird.

<packet version=”1.6.2.0”>

<server>

<lic_install>
<license>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiID8+IA0KLSA8cHAtc2
1iLXVuaXgtMTA6a2V5IGNvcmU6Zm9ybWF0PSJvcGVuZnVzaW9uLTMiIHhtbG5zOmNvcm
U9Imh0dHA6Ly9wYXJhbGxlbHMuY29tL3NjaGVtYXMva2V5cy9jb3JlLzMiIHhtbG5zOnBwLX
NtYi11bml4LTEwPSJodHRwOi8vcGFyYWxsZWxzLmNvbS9zY2hlbWFzL2tleXMvcHJvZHVjd
HMvcHAtc21iL3VuaXgvMTAuMCI+DQotIDwhLS0gVW5pcXVlIHByb2R1Y3QgS2V5IG51bWJ
lcg0KICAtLT4gDQogIDxjb3JlOmtleS1udW1iZXIgY29yZTp0eXBlPSJzdHJpbmciPlNNQi4wM
DAwMDAwMDwvY29yZTprZXktbnVtYmVyPiANCi0gPCEtLSBLZXkgZGVzY3JpcHRpb24NCiAgLS0+IA0KICA8cHAtc2
1iLXVuaXgtMTA6a2V5IGNvcmU6dHlwZT0ic3RyaW5nIj4wMDAwPC9jb3JlOmtl
eS12ZXJzaW9uIGNvcmU6dHlwZT0ic3RyaW5nIj4wLjA8L2NvcmU6dmVyc2lvbj4NCiAgPC
pkZXNjcmlwdGlvbj4NCiAgPGNvcmU6a2V5dHlwZT5QYXJhbGxlbHMgUGFuZWwgMTAgU0
1CIEVkaXRpb24gZm9yIFVuaXgvTGludXg8L2NvcmU6a2V5dHlwZT4gDQogIDwvY29yZTpk
ZXNjcmlwdGlvbj4NCi0gPCEtLSBQcm9kdWN0IHdoaWNoIHRoaXMgbGljZW5zZSBpcyBpbn
RlbmRlZCB0byB3b3JrIG9uDQogIC0tPiANCiAgPGNvcmU6cHJvZHVjdCBjb3JlOnR5cGU9In
N0cmluZyI+cHAtc21iL3VuaXg8L2NvcmU6cHJvZHVjdD4gDQotIDwhLS0gU3VwcG9ydGU
gdHlwZT0iYm9vbGVhbiI+dHJ1ZTwvY29yZTp0eXBlPSJzdH
JpbmciPjEwLjA8L2NvcmU6dmVyc2lvbj4gDQogIDxjb3JlOnZlcnNpb24gY29yZTp0eXBlPSJzdH
JpbmciPjEwLjA8L2NvcmU6dmVyc2lvbj4gDQogIDwhLS0gRGF0ZSBhZnRlciB3aGljaCB0aGlz
IGxpY2Vuc2UgYmVjb21lcyB1c2FibGUgKGluc2l2ZSkNCiAgLS0+IA0KICA8cHAtc2
1iLXVuaXgtMTA6a2V5IGNvcmU6dHlwZT0iZGF0ZSI+aW5zdGFudDwvY29yZTprZXktbnVt
YmVyPiANCi0gPCEtLSBEYXRlIGJlZm9yZSB3aGljaCB0aGlzIGxpY2Vuc2UgaXMgdXNhYmxlIChleGN
sdXNpdmUpDQogIC0tPiANCiAgPGNvcmU6ZXhwaXJhdGlvbi1kYXRlIGNvcmU6dHlwZT0iZ
GF0ZSI+bmV2ZXI8L2NvcmU6ZXhwaXJhdGlvbi1kYXRlPiANCi0gPCEtLSBVUkwgb2YgdGhlI
HNlcnZpY2UgZW5kcG9pbnQgdG8gdXNlIHdoZW4gcGVyZm9ybWluZyBhbiBhdXRvdXBkYX
RlDQogIC0tPiANCiAgPGNvcmU6bGljZW5zZS1zZXJ2ZXItdXJsIGNvcmU6dHlwZT0ic3RyaW
5nIj5odHRwczovL2thLnBhcmFsbGVscy5jb206NTIyNC94bWxycGM8L2NvcmU6bGljZW5zZS
1zZXJ2ZXItdXJsPiANCi0gPCEtLSBEYXRlIHdoZW4gcHJvZHVjdCB3aWxsIHRyeSB0byBwZ
XJmb3JtIGFuIGF1dG91cGRhdGUNCiAgLS0+IA0KICA8cHAtc2VydmVyLWJhc2Ut

VyZSB4bWxuczpkc2lnPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIj4NCi
0gPGRzaWc6U2lnbmVkSW5mbz4NCiAgPGRzaWc6Q2Fub25pY2FsaXphdGlvbk1ldGhvZC
BBbGdvcml0aG09Imh0dHA6Ly9wYXJhbGxlbHMuY29tL3NjaGVtYXMva2V5cy9jb3JlLzMjY2
Fub25pY2FsaXplIiAvPiANCiAgPGRzaWc6U2lnbmF0dXJlTWV0aG9kIEFsZ29yaXRobT0iaH
R0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxkc2lnLW1vcmUjcnNhLXNoYTI1NiIgLz4gD
QotIDxkc2lnOlJlZmVyZW5jZSBVUkk9IiI+DQotIDxkc2lnOlRyYW5zZm9ybXM+DQogIDxkc2ln
OlRyYW5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnL1RSLzIwMDEvUkVDL
XhtbC1jMTRuLTIwMDEwMzE1I1dpdGhDb21tZW50cyIgLz4gDQogIDxkc2lnOlRyYW5zZm9y
bSBBbGdvcml0aG09Imh0dHA6Ly9wYXJhbGxlbHMuY29tL3NjaGVtYXMva2V5cy9jb3JlLzMj
dHJhbnNmb3JtIiAvPiANCiAgPC9kc2lnOlRyYW5zZm9ybXM+DQogIDxkc2lnOkRpZ2VzdE1ld
GhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3NoYTI
1NiIgLz4gDQogIDxkc2lnOkRpZ2VzdFZhbHVlPlRrUzk2YnlycEpVdTVpUTdIbkhZYUhhMTIra
2lPYmZhRXBYMC9TWU5iV2M9PC9kc2lnOkRpZ2VzdFZhbHVlPiANCiAgPC9kc2lnOlJlZmV
yZW5jZT4NCiAgPC9kc2lnOlNpZ25lZEluZm8+DQogIDxkc2lnOlNpZ25hdHVyZVZhbHVlPmZ
vcUxYVFErTWhRUm44Y2NzSmdoU1ZhNUJUVi9kaUtaVGJjdk9qUnBXcW5XaDlXei9NQmF
KY0pDNnFnZUNkMHJ4OFZsWERxcm0yVUYgS1I1d0dWeFM1ZSt6NWNqREFEcVUyRStG
OXRaUndubXRuamZ4dHV0THZibjVyS0FYc1VtWTQ2OU1EcjNQeFRYbi9mYjZycmpvbGZO
OCBOWVE1WUVtMDkyd3IrWTg2TGNIVkhKM2ZXK01sQWd3WXpua0lzdFVQSE1JMVVkYl
gwZWkvWTFmcStSc0dSSnNhZ2dRNW56MzgralV5IFRvUDBRK2VrUnAxcUFVa09OWUYy
M21FT1FuQ2V5eFNqQ3FrejNyZEhkaVFFQTBKcmhYMGowYWZYWDJkL2JRUnZVeU93U
FBsVWhzNU4gY3FjcndRRDdMQ0lsOE9Udk5FaFRrNFo0OGdaM1ZIRzJ2bitpVGc9PTwvZH
NpZzpTaWduYXR1cmVWYWx1ZT4gDQotIDxkc2lnOktleUluZm8+DQotIDxkc2lnOlg1MDlEY
XRhPg0KICA8ZHNpZzpYNTA5Q2VydGlmaWNhdGU+TUlJRW5EQ0NBb1FDQVJFd0RRW
UpLb1pJaHZjTkFRRUZCUUF3Z2FZeEN6QUpCZ05WQkFZVEFrSk5NUXN3Q1FZRFZRUUl
Fd0pJVFRFUiBNQThHQTFVRUJ4TUlTR0Z0YVd4MGIyNHhIREFhQmdOVkJBb1RFMU5YY
zI5bWRDQkliMnhrYVc1bmN5Qk1kR1F4SFRBYkJnTlZCQXNUIEZFbHVkR1Z5Ym1Gc0lFU
mxkbVZzYjNCdFpXNTBNUnd3R2dZRFZRUURFeE5MUVNCeWIyOTBJR05sY25ScFptbGp
ZWFJsTVJ3d0dnWUogS29aSWh2Y05BUWtCRmcxclVQnpkM052Wm5RdVkyOXRNQjRYR
FRBNU1EWXdNakEyTURrek0xb1hEVE0yTVRBeE9EQTJNRGt6TTFvdyBnWUF4Q3pBSkJ
nTlZCQVlUQWtKTk1Rc3dDUVlEVlFRSUV3SklUVEVSTUE4R0ExVUVCeE1JU0dGdGFXeD
BiMjR4SERBYUJnTlZCQW9UIEUxTlhjMjltZENCSWIyeGthVzVuY3lCTWRHUXhIVEFiQmdO
VkJBc1RGRWx1ZEdWeWJtRnNJRVJsZG1Wc2IzQnRaVzUwTVJRd0VnWUQgVlFRREV3d
HdjQzF6YldJdGRXNXBlRENDQVNJd0RRWUpLb1pJaHZjTkFRUJCUUFEZ2dFUEFEQ0N
BUW9DZ2dFQkFNYzR0RWVhRFMzciBDS0sxQ1RaUm9aMTdGVzU5Z0tBbTlOQmt6V3Zw
ZW9VM0lnZU5SMTMrdWs0My9lYW1EQkNHZTBtQnNwb05vWjErWDk3RXVGYzB4b2RvIG
g0NVgrRUNIZk5kME1adzVrSFljWWF1M2kxSVBBakZMdm5DcE52ZXc1WHpXQzcwZ0xlen
ZNaWxRbWxTYWpLSEFEL0EvMGhzYkI0aDkgeVdxNTZPcDRiT1hUTDVKN0NZV1JHeVRS
aFV1MnFPbjZ2Y2o3Q2Rkb05oeWhRdDJTYlArNDlVMXFJdjMvOFZ3YnZrMzZhTVlTVlBlMS
BHRk50MlE3RkhKUUxicW1pY1doL0JBWkpjU2FGMWlPSUhuZ2VIbG9uSGptbzdJWUEvSX
cyTUZMT0dPam82Kzg2M0dUbDJUOXA0dGxGIHZHNWhOMWFpNHkybDdOVzFTczhVT1J
SclVlMENBd0VBQVRBTkJna3Foa2lHOXcwQkFRVUZBQU9DQWdFQUgrcnFFdkhNdEFRa
StWOHEgSDdFM3VSRk8xaVA1QzBVejlDQVdMZ0QyNWthWTlSaUV1K1ZNcGtyZ2RwMkR
KZUxIaWd5Q3MyM3dUbkZ6SS9qMUJNRjJ1VGdKS1B6MiBSOTA2YVcyR0tJWWpIencvd3
BmUDUxck9CWTVQNlhpdXNtM2NGdk8zTnJ4Z0hJSy9WV3N5WW1EN25mMksyZUptQmsv
ZU85TklaalBoIGRkS2NrYmNZRVhRN3dzd2l0dDlYcTFuckg2eFNXT1lIVUUwcHI0QW1jVXc3
TDdRQVUzdDBVM0lIQzNHYmVyVE1RNERSL2xDTFV5a24gUkl3aDNKQXZkYVFJZnA2TW
9yNkY0VWtrMWh2bzkwVTR1VHFEbXc4RFJHcVRoOCtPUEY1V1dTL0MyK0o0bThOWTND
WjZpa3FIc0FpcCB5Si9uR1pqU3cvdHc4N0xMWHBMMzJhRkFLcGpjOGc1MkZuc3YzVDlwdj
RHd0Q2V0hHYU1DNUVsL21XSWplSE5pLytPTk5hbkdHbHJxIHVLQ1p2UktHenJRNjV4bHIy
eEovTXFTWDV3cWdTK3R5SE1XY3BwVFpZdmF6cXZtZFRUbllNWlhwd1M1YWxqQlAwb3F
Wd09wVDFLUksgY0o3bXVaYzNUWlJ1OHY0b2pWa1dWTlF2QVNUUFNsdXFkcW42RFdw
V2V3UEQ0ZXg3c2ZJam52N1phWmtZV2tPL3VDam9SeE4wbmFtVSBlVk5QemZSNXJ5em
Vjalozb0g0L1NXTGFYeHFiQ2k0QzBjMFlVeTZBMkZkejVKK3RwWUdFbkVFMENpcDFyRjVn
MFJhMURBZXV4ZFFaIEMwang3NlNsQ1VjSWhGQ0p4RmRxMk5JMHZJVnlVNTl6R2NrY3E

3WVZZVlJ2czNVc2crb1pNRTBKbTFjPTwvZHNpZzpYNTA5Q2VydGlmaWNhdGU+IA0KICA
8L2RzaWc6WDUwOURhdGE+DQogIDwvZHNpZzpLZXlJbmZvPg0KICA8L2RzaWc6U2lnbm
F0dXJlPg0KICA8L3BwLXNtYi11bml4LTEwOmtleT4=</license>

</lic_install>

</server>

</packet>

Dieses Abfrage-Paket installiert einen zusätzlichen Lizenzschlüssel.

Bitte beachten Sie, dass der Inhalt des **license**-Knotens ein gültiger base64-Code ist, aber kein Lizenzschlüssel. Wenn Sie das folgende Abfrage-Paket ausgeben, erhalten Sie eine Fehlermeldung “error 1020”.

<packet version=”1.6.2.0”>

<server>

<lic_install>
<license>TGV0IHVzIGp1c3QgcHJldGVuZCB0aGF0IHRoaXMgaXMgYSBQYXJhbGxlbHMg
UGFuZWwgbGljZW5zZSBrZXkuCk9mIGNvdXJzZSwgd2UgY291bGQgbm90IGluY2x1ZGUg
YSByZWFsIGxpY2Vuc2Uga2V5IGluIHRoZSBkb2N1bWVudGF0aW9uLgpUaGFuayB5b3Ug
Zm9yIHVuZGVyc3RhbmRpbmcu</license>

<additional_key/>

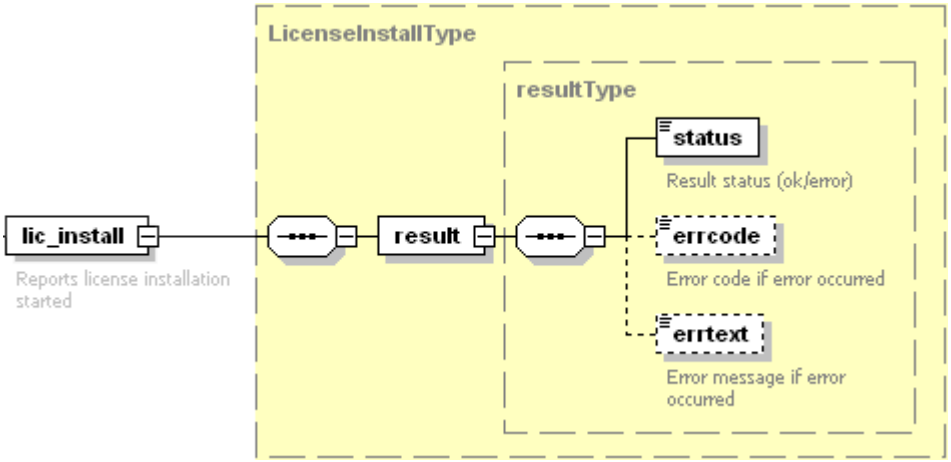
</lic_install>

</server>

</packet>

Struktur des Antwortpakets

Der **lic_install**-Knoten des Antwortpakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort.
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string.* Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *unsignedInt.*
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string.*

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<server>
<lic_install>
    <result>
    <status>ok</status>
    </result>
</lic_install>
</server>
</packet>
```

Sie erhalten diesen Fehler, falls versucht wird, über das Abfrage-Paket einen ungültigen Lizenzschlüssel zu installieren.

```
<packet version=”1.6.2.0”>
<server>
    <lic_install>
    <result>
        <status>error</status>
        <errcode>1020</errcode>
  <errtext>The uploaded key file is not valid or does not contain a license key.</errtext>
    </result>
    </lic_install>
</server>
</packet>
```

DNS-Konfiguration

Um den DNS-Dienst des Panels über das API RPC-Protokoll zu installieren, verwenden Sie bitte den **dns**-Operator. Gehen Sie hierzu wie im Folgenden beschrieben vor.

In diesem Abschnitt:

DNS-Dienst deaktivieren 100
Ändern des SOA-Record-Templates 102
Einrichten von Templates für Ressourceneinträge 106

DNS-Dienst deaktivieren

Um den DNS-Dienst des Panels per API RPC zu deaktivieren, erstellen Sie ein XML-Abfrage-Paket, das den **dns/disable**-Kontrollknoten enthält:

<packet version=”1.6.2.0”>

<dns>

<disable/>

</dns>

</packet>

In diesem Abschnitt:

Abfrage-Paket 100
Antwortpaket 101

Abfrage-Paket

Das folgende Abfrage-Paket deaktiviert den DNS-Dienst:

<packet version=”1.6.2.0”>

<dns>

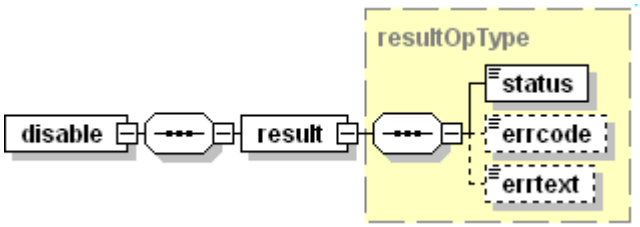
<disable/>

</dns>

</packet>

Antwortpaket

Der **disable**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.

Beispiel

Eine positive Antwort sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<dns>
<disable>
<result>
  <status>ok</status>
</result>
</disable>
</dns>
</packet>
```

Ändern des SOA-Record-Templates

Um ein neues SOA-Record-Template einzurichten, erstellen Sie ein XML-Abfrage-Paket, das den **dns/set**-Kontrollknoten enthält:

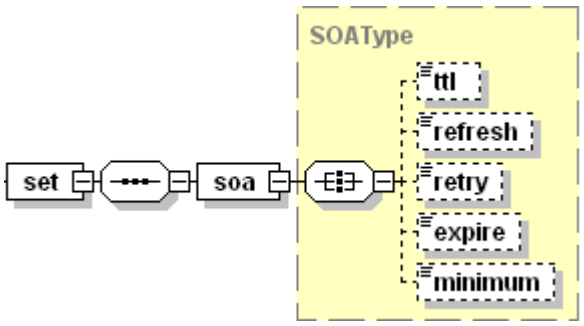
```
<packet version=”1.6.2.0”>
<dns>
<set>
  . . .
</set>
</dns>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets 102
Beispiele für Abfragen 103
Struktur des Antwortpakets 104
Antwort-Beispiele 105

Struktur des Abfrage-Pakets

Um ein neues SOA-Record-Template einzurichten, erstellen Sie ein neues XML-Abfrage-Paket, das den **dns/set**-Kontrollknoten enthält und wie folgt strukturiert ist:



- **soa**, *erforderlich*
  Gibt die SOA-Parameter an. Datentyp: *SOAType* (`plesk_dns.xsd`).
  - **ttl**, *optional*
    Gibt den Parameterwert `ttl` an. Das Panel legt als Standardwert einen Tag fest. Datentyp: *unsignedInt.*
  - **refresh**, *optional*
    Gibt den Parameterwert `refresh` an. Das Panel legt als Standardwert drei Stunden fest. Datentyp: *unsignedInt.*
  - **retry**, *optional*
    Gibt den Parameterwert `retry` an. Das Panel legt als Standardwert eine Stunde fest. Datentyp: *unsignedInt.*
  - **expire**, *optional*
    Gibt den Parameterwert `expire` an. (Signiert 32-Bit-Wert in Sekunden.)
    Das Panel legt als Standardwert eine Woche fest. Datentyp: *unsignedInt.*
  - **minimum**, *optional*
    Gibt den Parameterwert `minimum` an. Das Panel legt als Standardwert drei Stunden fest. Datentyp: *unsignedInt.*

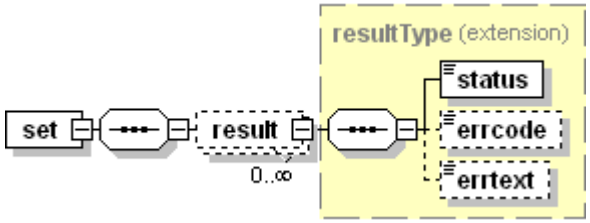
Beispiele für Abfragen

Dieses Paket aktualisiert ein SOA-Record-Template, so dass es den TTL (Time to Live)-Intervall auf 32 Stunden einstellt, den Aktualisierungs-(Refresh) Intervall auf 6 Stunden und den Erneut versuchen-(Retry) Intervall auf 15 Minuten. Die Panel-Standardwerte für die Intervalle Ablaufen (Expire) und Minimum werden unverändert beibehalten.

```
<packet version=”1.6.2.0”>
<dns>
<set>
    <soa>
    <ttl>115200</ttl>
    <refresh>21600</refresh>
    <retry>900</retry>
    </soa>
</set>
</dns>
</packet>
```

Struktur des Antwortpakets

Der **set**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

<packet version=”1.6.2.0”>
<dns>
<set>
<result>
<status>ok</status>
</result>
</set>
</dns>
</packet>

Die folgende negative Antwort wird empfangen, falls ein Wert viel zu groß ist:

ABFRAGE-PAKET

<packet version=”1.6.2.0”>
<dns>
<set>
<soa>
<refresh>4320000000</refresh>
</soa>
</set>
</dns>
</packet>
ANTWORTPAKET

<packet version=”1.6.2.0”>

<system>
<status>error</status>
<errcode>1014</errcode>
<errtext>Parser error: Request is invalid</errtext>
</system> </packet>

Einrichten von Templates für Ressourceneinträge

Über API RPC können Sie Ressourceneinträge in dem DNS-Zone-Template auf folgende Weise verändern:

- neue Ressourceneinträge hinzufügen
- existierende Ressourceneinträge entfernen (es ist möglich eine Liste mit den Ressourceneinträge (Akk. RR = Resource Records) abzurufen, die bereits in dem Template enthalten sind)

In diesem Abschnitt:

Hinzufügen eines Templates für Ressourceneinträge 106
Informationen über das DNS-Zonen-Template abrufen 115
Entfernen eines Templates für Ressourceneinträge 121

Hinzufügen eines Templates für Ressourceneinträge

Um ein Template für DNS-Zonen- Ressourceneinträge hinzufügen, erstellen Sie ein XML-Abfrage-Paket **dns/add-rec**-Kontrollknoten:

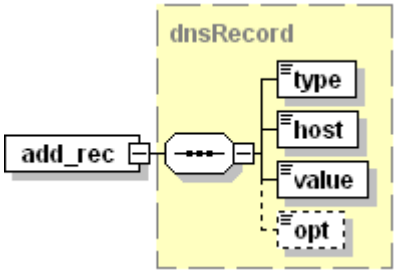
```
<packet version="1.6.2.0">
<dns>
<add_rec>
  . . .
</add_rec>
</dns>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets..... 107
Beispiele für Abfragen 108
Struktur des Antwortpakets 113
Antwort-Beispiele 114

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket, mit dem ein neuer DNS-Eintrag zu dem DNS-Zone-Template hinzugefügt wird, beinhaltet den Kontrollknoten **dns/add_rec**, der durch den *dnsRecord* type (`plesk_dns.xsd`) dargestellt wird und wie folgt strukturiert ist:



- **type**, *erforderlich*
  Gibt den Typ des DNS-Eintrags an. Datentyp: *string*.
  Zulässige Werte: A | NS | CNAME | MX | PTR | TXT | SOA | AXFR | SRV.
- **host**, *erforderlich*
  Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden. Datentyp: *string.*
- **value**, *erforderlich*
  Gibt den Wert an, der mit dem Wert **host** verknüpft wird. Datentyp: *string.*
  Der Wert enthält die Platzhalter `<domain>` oder `<ip>`, die durch echte IP-Adressen/Hostnamen ersetzt werden, wenn eine DNS-Zone nach der Erstellung der Domain erstellt wird.
- **opt**, *optional*
  Beinhaltet optionale Informationen über den DNS-Eintrag. Datentyp: *string.*

Hinweis: Im Falle des SRV-Eintrags kann der **opt** -Knoten zusätzlichen XML-Code in dem folgenden Format enthalten:`<Srv Protocol=”” Port=”” Priority=”” Weight=””/>`.

Sie können mehrere DNS-Einträge über ein einziges Paket hinzufügen. Fügen Sie so viele <add-rec>-Operationen hinzu, wie Sie DNS-Einträge hinzufügen wollen.

<dns>

```
  <add_rec>
  ...
  </add_rec>
...
  <add_rec>
  ...
  </add_rec>
```

</dns>

Beispiele für Abfragen

NS-Record-Template

```
<packet version="1.6.2.0">
<dns>
    <add_rec>
    <type>NS</type>
    <host/>
    <value>ns.&lt;domain&gt;</value>
    </add_rec>
</dns>
</packet>
```

DNS-Zone erstellt auf *example.com*-Domainerstellung wird den folgenden Ressourceneintrag (RR) enthalten:

```
example.com.   NS   ns.example.com.
```

macht `ns.example.com` zum Nameserver des Hosts `example.com`.

Das Abfrage-Paket fügt zu dem DNS-Zonen-Template einen Eintrag hinzu, der `provider-dns-server.example.com` zu einem Nameserver für eine Domain macht.

```
<packet version="1.6.2.0">
<dns>
    <add_rec>
    <type>NS</type>
    <host/>
  <value>provider-dns-server.example.com</value>
  </add_rec>
</dns>
</packet>
```

Ein Record-Template

```
<packet version="1.6.2.0">
<dns>
    <add_rec>
    <type>A</type>
    <host>mail</host>
```

<value><ip></value>

</add_rec>

</dns>

</packet>

DNS-Zone erstellt auf *example.com*-Domainerstellung (gehostet auf IP 192.0.2.12) wird den folgenden Ressourceneintrag (RR) enthalten:

mail.example.com. A 192.0.2.12

definiert die IP-Adresse `192.0.2.12` für den Host `mail.example.com`.

CNAME-Record-Template

<packet version="1.6.2.0">

<dns>

<add_rec>

<type>CNAME</type>

<host>www</host>

<value>ns.<domain></value>

</add_rec>

</dns>

</packet>

DNS-Zone erstellt auf *example.com*-Domainerstellung wird den folgenden Ressourceneintrag (RR) enthalten:

www NS ns

macht `ns.example.com` zum Alias für `example.com`.

MX-Record-Template

<packet version="1.6.2.0">

<dns>

<add_rec>

<type>MX</type>

<host/>

<value>exchange.<domain></value>

<opt>0</opt>

</add_rec>

</dns>

</packet>

DNS-Zone erstellt auf *example.com*-Domainerstellung wird den folgenden Ressourceneintrag (RR) enthalten:

example.com. MX 0 exchange.example.com.

macht `mail.example.com` zum wichtigsten Mailserver für die Domain `example.com`.

PTR-Record-Template

```
<packet version="1.6.2.0">
<dns>
  <add_rec>
    <type>PTR</type>
    <host>&lt;ip&gt;</host>
    <value>community</value>
    <opt>24</opt>
  </add_rec>
</dns>
</packet>
```

DNS-Zone erstellt auf *example.com*-Domainerstellung (gehostet auf IP 192.0.2.12) wird den folgenden Ressourceneintrag (RR) enthalten:

192.0.2.12/24 PTR community.example.com

macht die Domain `community.example.com` zu dem Domain-Pointer für das Subnetz `192.0.2.12/24`.

TXT-Record-Template

```
<packet version="1.6.2.0">
<dns>
  <add_rec>
    <type>TXT</type>
    <host>about</host>
    <value>Delivered by the best provider</value>
  </add_rec>
</dns>
</packet>
```

DNS-Zone erstellt auf *example.com*-Domainerstellung wird den folgenden Ressourceneintrag (RR) enthalten:

about TXT Delivered by the best provider

fügt eine Textbeschreibung “Delivered by the best provider” zu der Domain `about.example.com` hinzu.

SRV-Record-Template

```
<packet version=”1.6.2.0”>
<dns>
  <add_rec>
    <type>SRV</type>
    <host>_LDAP._tcp</host>
    <value>192.0.2.4.</value>
    <opt>5 25 220</opt>
  </add_rec>
</dns>
</packet>
```

DNS-Zone erstellt auf example.com-Domainerstellung wird den folgenden Ressourceneintrag (RR) enthalten:

```
_LDAP._tcp      SRV 5 25 220 192.0.2.4.
```

bringt einen Client des `LDAP`-Dienstes dazu, auf der Domain `example.com` eine SRV-Suche von `192.0.2.4` durchzuführen.

Mehrere Einträge (Records) hinzufügen

Das folgende Abfrage-Paket fügt alle Einträge aus den obigen Beispielen auf einmal hinzu:

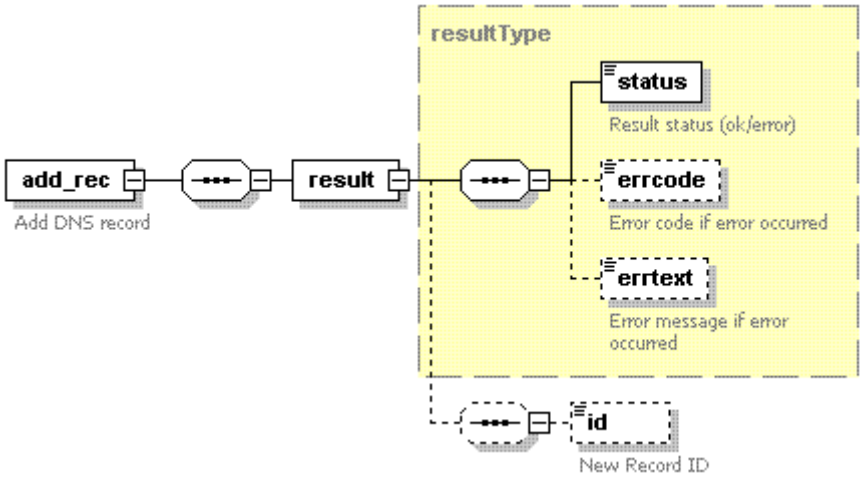
```
<packet version=”1.6.2.0”>
<dns>
    <add_rec>
    <type>NS</type>
    <host/>
    <value>ns.&lt;domain&gt;</value>
    </add_rec>

    <add_rec>
    <type>NS</type>
    <host/>
  <value>provider-dns-server.example.com</value>
  </add_rec>
    <add_rec>
```

```
    <type>A</type>
    <host>mail</host>
    <value>&lt;ip&gt;</value>
    </add_rec>
  <add_rec>
    <type>CNAME</type>
    <host>www</host>
    <value>ns.&lt;domain&gt;</value>
  </add_rec>
  <add_rec>
    <type>MX</type>
    <host/>
    <value>exchange.&lt;domain&gt;</value>
    <opt>0</opt>
  </add_rec>
  <add_rec>
    <type>PTR</type>
    <host>&lt;ip&gt;</host>
    <value>community</value>
    <opt>24</opt>
  </add_rec>
  <add_rec>
    <type>TXT</type>
    <host>about</host>
    <value>Delivered by the best provider</value>
  </add_rec>
  <add_rec>
    <type>SRV</type>
    <host>_LDAP._tcp</host>
    <value>192.0.2.4.</value>
    <opt>5 25 220</opt>
  </add_rec>
</dns>
</packet>
```

Struktur des Antwortpakets

Der **add_rec**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **id**, *erforderlich,* falls die Operation erfolgreich abgeschlossen wird.
  Liefert den eindeutigen Identifier des gerade hinzugefügten DNS-Eintrags. Datentyp: *integer*.

Antwort-Beispiele

Eine positive Antwort sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<dns>
<add_rec>
<result>
    <status>ok</status>
    <id>17</id>
</result></add_rec>
</dns>
</packet>
```

Folgende negative Antwort wird erhalten, falls ein Abfrage-Paket unzulässige Werte für den Ressourceneintrag (Resource Record) enthält, der hinzugefügt werden soll (in unserem Beispiel handelt es sich um einen falsch analysierten und angegebenen Host für den SRV-Eintrag):

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<dns>
<add_rec>
  <type>SRV</type>
  <host>_LDAP._tcp.ldap.domain-test-480908606.tst.</host>
    <value>192.0.2.4.</value>
    <opt>5 25 220</opt>
</add_rec>
</dns>
</packet>
```

ANTWORTPAKET

```
<packet version=”1.6.2.0”>
<dns>
    <add_rec>
    <result>
        <status>error</status>
        <errcode>1019</errcode>
     <errtext>Incorrect DNS record values were specified.</errtext>
```

```
    </result>
    </add_rec>
</dns>
</packet>
```

Informationen über das DNS-Zonen-Template abrufen

Um Informationen über die Ressourceneinträge in dem DNS-Zonen-Template abzurufen, erstellen Sie ein XML-Paket mit dem **dns/get_rec**-Kontrollknoten:

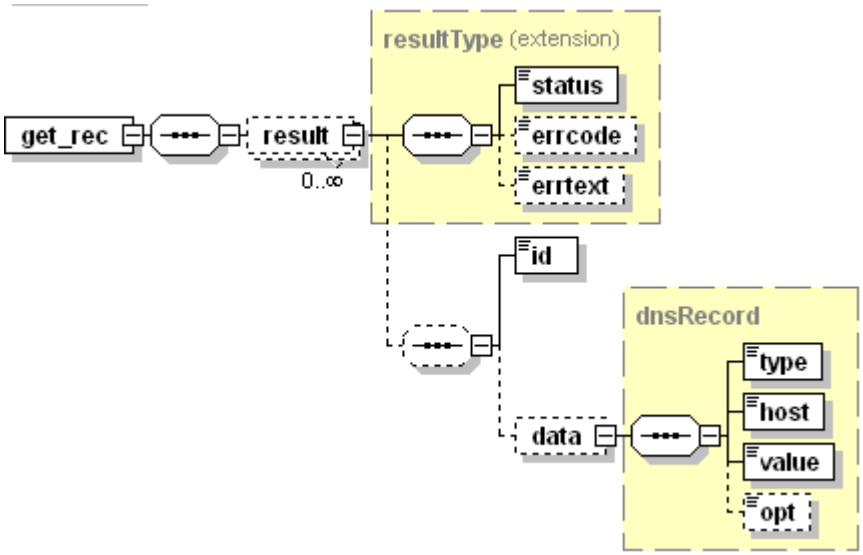
```
<packet version="1.6.2.0">
<dns>
<get_rec>
    <filter/>
    <template/>
</get_rec>
</dns>
</packet>
```

In diesem Abschnitt:

Struktur des Antwortpakets 116
Antwort-Beispiele 118

Struktur des Antwortpakets

Der **get_rec**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **id**, *erforderlich,* falls die Operation erfolgreich abgeschlossen wird.
  Liefert den eindeutigen Identifier des DNS-Eintrags. Datentyp: *integer*.
- **data***, erforderlich,* falls die Operation erfolgreich abgeschlossen wird.
  Beinhaltet eine Reihe von Ressourceneintragsdaten.
  - **type**, *erforderlich*
    Gibt den Typ des DNS-Eintrags an. Datentyp: *string*.
    Zulässige Werte: A | NS | CNAME | MX | PTR | TXT | SOA | AXFR | SRV.
  - **host**, *erforderlich*
    Gibt den Namen des Hosts an, der vom DNS verwendet wird. Datentyp: *string.*
  - **value**, *erforderlich*
    Gibt den Wert an, der mit dem Wert **host** verknüpft wird. Datentyp: *string.*
    Der Wert enthält die Platzhalter `<domain>` oder `<ip>`, die durch echte IP-Adressen/Hostnamen ersetzt werden, wenn eine DNS-Zone nach der Erstellung der Domain erstellt wird.
  - **opt**, *optional*
    Beinhaltet optionale Informationen über den DNS-Eintrag. Datentyp: *string.*

Antwort-Beispiele

Die folgende Antwort wird empfangen, falls das DNS-Zonen-Template keinen einzigen Ressourceneintrag enthält:

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<dns>
<get_rec>
    <filter/>
    <template/>
</get_rec>
</dns>
</packet>
```

ANTWORTPAKET

```
<packet version=”1.6.2.0”>
<dns>
  <get_rec/>
</dns>
</packet>
```

Die folgende Antwort enthält eine Beschreibung des Standard-DNS-Zonen-Templates des Panels:

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<dns>
<get_rec>
    <filter/>
    <template/>
</get_rec>
</dns>
</packet>
```

ANTWORTPAKET

```
<packet version=”1.6.2.0”>
<dns>
```

```
<get_rec>
<result>
    <status>ok</status>
    <id>1</id>
    <data>
    <type>NS</type>
    <host>&lt;domain&gt;.</host>
    <value>ns.&lt;domain&gt;.</value>
    <opt/>
    </data>
</result>
<result>
    <status>ok</status>
    <id>2</id>
    <data>
    <type>A</type>
    <host>ns.&lt;domain&gt;.</host>
    <value>&lt;ip&gt;</value>
    <opt/>
    </data>
</result>
<result>
    <status>ok</status>
    <id>3</id>
    <data>
    <type>A</type>
    <host>&lt;domain&gt;.</host>
    <value>&lt;ip&gt;</value>
    <opt/>
    </data>
</result>
<result>
    <status>ok</status>
    <id>4</id>
    <data>
    <type>A</type>
    <host>webmail.&lt;domain&gt;.</host>
```

```
        <value>&lt;ip&gt;</value>
        <opt/>
        </data>
    </result>
    <result>
        <status>ok</status>
        <id>5</id>
        <data>
        <type>MX</type>
        <host>&lt;domain&gt;.</host>
        <value>mail.&lt;domain&gt;.</value>
        <opt>10</opt>
        </data>
    </result>
    <result>
        <status>ok</status>
        <id>6</id>
        <data>
        <type>A</type>
        <host>mail.&lt;domain&gt;.</host>
        <value>&lt;ip&gt;</value>
        <opt/>
        </data>
    </result>
    <result>
        <status>ok</status>
        <id>7</id>
        <data>
        <type>CNAME</type>
        <host>ftp.&lt;domain&gt;.</host>
        <value>&lt;domain&gt;.</value>
        <opt/>
        </data>
    </result>
    <result>
        <status>ok</status>
        <id>8</id>
```

```
        <data>
        <type>PTR</type>
        <host>&lt;ip&gt; / 24</host>
        <value>&lt;domain&gt;.</value>
        <opt>24</opt>
        </data>
    </result>
    </get_rec>
</dns>
</packet>
```

Entfernen eines Templates für Ressourceneinträge

Um einen Ressourceneintrag (Resource Record) von dem DNS-Zonen-Template zu entfernen, erstellen Sie ein XML-Paket, das den **dns/del_rec**-Kontrollknoten enthält:

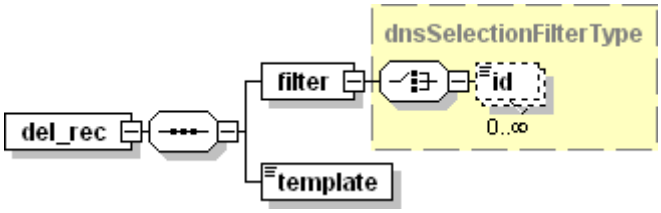
```
<packet version="1.6.2.0">
<dns>
<del_rec>
  . . .
</del_rec>
</dns>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets 122
Beispiele für Abfragen 123
Struktur des Antwortpakets 124
Antwort-Beispiele 125

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket über das ein DNS-Eintrag gelöscht wird, beinhaltet den **del_rec**-Kontrollknoten, der wie folgt strukturiert ist:



- **filter**, *erforderlich*

  Gibt die Filterregel an. Datentyp: *dnsSelectionFilterType* (`dns_input.xsd`).

  Falls dieses Feld (`<filter/>`) leer ist, werden alle Record-Templates gelöscht.

  - **id**, *optional*

    Gibt die ID des Templates für den Ressourceneintrag an, das gelöscht werden soll. Datentyp: *id_type* (`common.xsd`).

- **template**, *erforderlich*

  Gibt die zu löschenden Record-Templates an. Datentyp: *none.*

Beispiele für Abfragen

Dieses Abfrage-Paket löscht aus dem DNS-Zonen-Template den Eintrag mit der ID 7.

```
<packet version=”1.6.2.0”>
<dns>
<del_rec>
    <filter>
    <id>7</id>
    </filter>
<template/>
</del_rec>
</dns>
</packet>
```

Dieses Abfrage-Paket löscht aus dem DNS-Zonen-Template die Einträge mit der ID 1-5.

```
<packet version=”1.6.2.0”>
<dns>
<del_rec>
    <filter>
    <id>1</id>
    <id>2</id>
    <id>3</id>
    <id>4</id>
    <id>5</id>
    </filter>
<template/>
</del_rec>
</dns>
</packet>
```

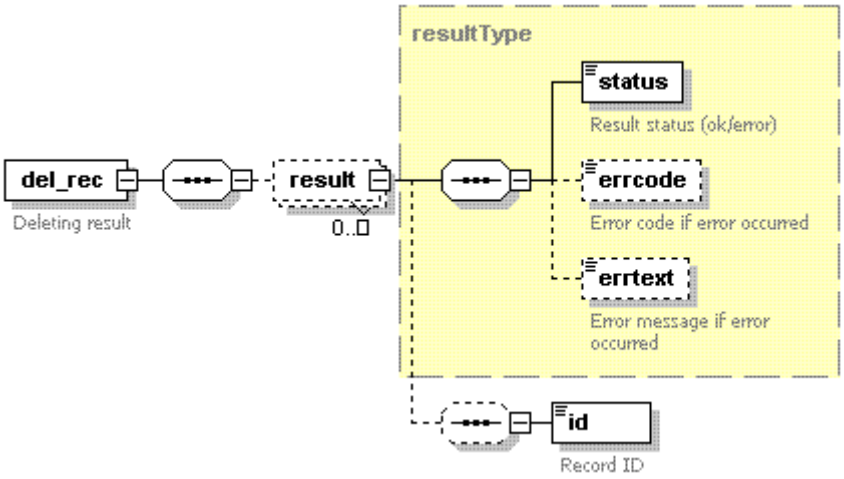
Dieses Paket entfernt alle Einträge aus dem DNS-Zonen-Template:

```
<packet version=”1.6.2.0”>
<dns>
<del_rec>
  <filter/>
```

<template/>
</del_rec>
</dns>
</packet>

Struktur des Antwortpakets

Der **del_rec**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **id**, *erforderlich, falls die Operation erfolgreich abgeschlossen wird.*
  Liefert den eindeutigen Identifier des entfernten DNS-Eintrags. Datentyp: *integer*.

Antwort-Beispiele

Eine positive Antwort, die vom Server empfangen wird (entfernt alle DNS-Zonen-Template-Einträge erfolgreich):

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<dns>
<del_rec>
    <filter/>
    <template/>
</del_rec>
</dns>
</packet>
ANTWORTPAKET
<packet version=”1.6.2.0”>
<dns>
    <del_rec>
    <result>
        <status>ok</status>
        <id>1</id>
    </result>
    <result>
        <status>ok</status>
        <id>2</id>
    </result>
    <result>
        <status>ok</status>
        <id>3</id>
    </result>
    <result>
        <status>ok</status>
        <id>4</id>
    </result>
    <result>
        <status>ok</status>
```

```
        <id>5</id>
    </result>
    <result>
        <status>ok</status>
        <id>6</id>
    </result>
    <result>
        <status>ok</status>
        <id>7</id>
    </result>
    <result>
        <status>ok</status>
        <id>8</id>
    </result>
    <result>
        <status>ok</status>
        <id>9</id>
    </result>
    <result>
        <status>ok</status>
        <id>10</id>
    </result>
    <result>
        <status>ok</status>
        <id>11</id>
    </result>
    <result>
        <status>ok</status>
        <id>12</id>
    </result>
    </del_rec>
</dns>
</packet>
```

Folgende negative Antwort wird empfangen, falls der angegebene, zu löschende Eintrag nicht existiert:

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<dns>
<del_rec>
<filter>
    <id>1</id>
    <id>2</id>
    <id>3</id>
    <id>4</id>
    <id>5</id>
</filter>
<template/>
</del_rec>
</dns>
</packet>
```

ANTWORTPAKET

```
<packet version=”1.6.2.0”>
<dns>
    <del_rec>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>DNS record does not exist.</errtext>
     <id>1</id>
    </result>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>DNS record does not exist.</errtext>
     <id>2</id>
    </result>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>DNS record does not exist.</errtext>
     <id>3</id>
```

```
    </result>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
      <errtext>DNS record does not exist.</errtext>
      <id>4</id>
    </result>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
      <errtext>DNS record does not exist.</errtext>
      <id>5</id>
    </result>
    </del_rec>
</dns>
</packet>
```

Installation von SSL-Zertifikaten

Um ein SSL-Zertifikat zu installieren, erstellen Sie ein XML-Abfrage-Paket mit dem **certificate/install**-Kontrollknoten:

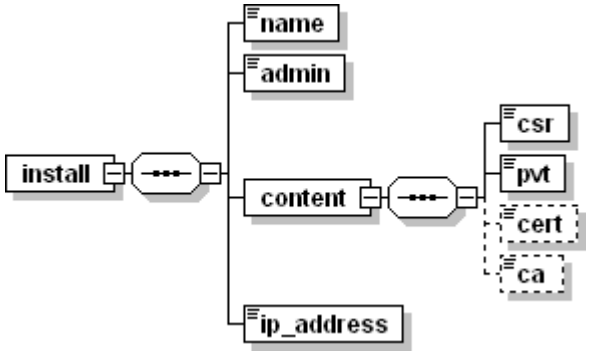
```
<packet version=”1.6.2.0”>
<certificate>
  <install>
    …
  </install>
</certificate>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets.....129
Abfragebeispiel 130
Struktur des Antwortpakets 133
Antwort-Beispiele 134

Struktur des Abfrage-Pakets

Der **install**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **name**, *erforderlich*
  Gibt den Namen an, unter dem das Zertifikat in dem Panel angezeigt wird. Datentyp: *string*.
- **admin**, *erforderlich*
  Gibt das Zertifikat an, das ins Server-Repository importiert wird. Datentyp: *none*.
- **content**, *erforderlich*
  Beinhaltet alle Daten, aus dem sich das Zertifikat zusammensetzt. Datentyp: none.
- **csr**, *erforderlich*
  Gibt das Zertifikat CSR an (Certificate Signing Request). Datentyp: *string*.
- **pvt**, *erforderlich*
  Gibt den privaten Schlüssel des Zertifikats an (Certificate Private Key). Datentyp: *string*.
- **cert**, *optional*
  Beinhaltet den Inhalt (Body) des Zertifikats. Datentyp: *string*.
- **ca**, *optional*
  Beinhaltet die Authentifizierungsinformationen des Zertifikats (Certificate Authority Body). Datentyp: *string*.
- **ip_address**, *erforderlich*
  Gibt die zu dem Zertifikat zugehörige IP-Adresse an. Datentyp: *ip_type* (`common.xsd`).

Abfragebeispiel

Dieses Paket installiert ein CA-Zertifikat (Certificate Authority), das in dem Panel unter dem Namen *common,* angezeigt wird und der IP-Adresse 192.0.2.14 zugewiesen wird.

<packet version="1.6.2.0">

<certificate>

<install>

<name>common</name>

<admin/>

<content>

<csr>-----BEGIN CERTIFICATE REQUEST-----

MIICwTCCAakCAQAwfDELMAkGA1UEBhMCVVMxEDAOBgNVBAgTB2dlb3JnaWExEDAO
BgNVBAcTB0F0bGFudGExEjAQBgNVBAoTCURvZSwgTHRkLjEUMBIGA1UEAxMLam9o
bmRvZS5vcmcxHzAdBgkqhkiG9w0BCQEWEGpkb2VAam9obmRvZS5vcmcwggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC7AQlX0pXqCs61ZUZ28zJ9PAG0yxvV
2pJnJTMqrudyi/3vV8IZrBUUWMthf7tQmMSSyZWAqM+7n4Q9Y4f5WuR+3ReM5kxC
hbQnOrjpdUmv1+ZhB9Q21UFA5eEptyDM/LR2JAjpKJD078rh76pVZm3EsJMl3+UE
3Dg0DeKDelDKa4SYzW1JJnoen4UXX37lIHdBnIj+NCQbHRSiGD6w6mMyiCMOzXCp
SBiGXgRRcaOVl50OeAdVKkwpJaae1K7K3q9j/LkAlOa1C8kuwdaCtKC+jFCE72I0
Q8TnIn7ahPy0yd2EgzZ42Ys8D3PziqGax8I/e6BbEOfzi0n1HD9GCzaPAgMBAAGg
ADANBgkqhkiG9w0BAQQFAAOCAQEAhoSVmInXLQDJu8jrZTVvFUx6O6aHOUkUf/G8
Wnc2x7w+Qo7XHMHVDCUcP3K+bEw9oKxfpvRXP9XTVhX2jABHrxywLyjg4cXwaUgR
t7ULvpLQ0Sum5XNtBypAdSQZZ7ktAa6Q3OMAFNdY8YG9J/L2zlcL9JJZLAUnRWzN
fMBn9Ip3pCC8N9OF9+PFpxGvhtlc1O27w9wj0RdK3hF6Rg37qcUJSnn6XxOB41Q8
ZZbX2Vbskz0WJ1IcxsfWDJ/HFdho7GrrD/wujNpwkTPc3xe2AHcfQX8c/92xyra/
kdzxsDdFD7IwA5bBWGm8QwlL9yWuDPiZ8vSGw96D/Qije9UVkA==
-----END CERTIFICATE REQUEST-----</csr>

<pvt>-----BEGIN RSA PRIVATE KEY-----

MIIEpQIBAAKCAQEAyxfczy4HqgwOI6yxEa+dLaMLe5zC0ijcawsha8oFtJlxv/DM
9INpdKjv9HaInrR0StjG9HgqTpYrhOCJxeB07/gGsQ82xsvbcKANuCSQKInpwVim
GUsisGNrfnbKIoAewN21ENCZQrufcQNWNzTx/GjVCXqa/Sy4pXJ6Jud0dzA/+Vis
SBw5Oo4IqI2MDvuoDk+exQLdgM45Vgq4d3A9+ESzmRmjmdovhNxnXdvMxA5RSrF+
GZlBTFftsLQsnfnNiMR6ZTFgNCt2vdDmUAV9iix2B1PuoH94TRCJRG5tVCFtfVzS
Rjo4ZzoiUs57efb3Oe56Lb64Ubi9ZRzNhdkGLwIDAQABAoIBABJdOAQr31mK+YRU
SzaUMJw9z/3/cMZcF4I+YWlDvTxVW6nFdmLttw1rcTcjuLrRCmryKxtT+k9c2NaN
DygrYaeJP+GmxIc5S8BlWJg8BiUEQ3TONUa4OozEkKXAzApOl3lNEuBQXGtiV8p6
SMN5MCBowkq3IWifMJsVPOJjr25Pby04HCPeeM+cnEiPnMrBjK4OGH8/pKqedf05
EcjnFKGFP2G0BUEhLX7ooMhB+AbjoIbGFXlBIzhaofs8zzSjNkKJ9i4LQMEe0GDf
zCgE4GehW+FdcUlhG1zO7FOJc0LdVUR4deoMzaiNRIwdWxbokVwgSEnM20yXYdCl
t+dsREECgYEA+sNPkzdQC0vF6me6WO8EqMu6AbPjObKnw0i8+g8ndRkMcJQ8FEXD
+D447fCMORUdqGFH8aMWDp392z4JX3+ksSN9XtyEEKB3fsknENsDZKVJvh9U+gUD
DO12mC1PleQBHBjN7Wlsg4mPZNv3cxUQLhFU0qW/moQdy0JrDfv+51ECgYEAz1Ww
OJolOjKhPkGxiDY9sh95E/rpuwkGjb0qiS/UFHHuLc9pDYHyKRpPggdT4nEuBz34
cRpihMgxBKEzuXGEPvdIST0aObOwcqfY4HAJ8PS08rZupCJsk413gYeI/ckXacQK
bYYHLKSEavrcmMFc98i2fmbbNp+ckeLHfzVYtX8CgYEAyP6X39YsEIHRx4sQ8IvU
3j89fnPjo7GxinPZFU3kQJWtROdsKIurAmVNWFrA6lgkh3xCIEqqOVklyv+0n5k+
NsXNjaWPLYyRe0xcRcRmudtKelu+zxAJW+lSb7OR4QD6arzvAmbIFb7C8wLlGpc9
es9lf9pe88kF4JACIxljPaECgYEAj4sWtwlZbsJwygZ3YAOVkUWi8QdNXLVx+R2X
XmVjokgCi2rGo5hszLIvi6mBFQwgvtjTsZJ/1Mg4z6i/g8sosONJA5OvHXXfWnIE

```
f9Zxu4Xf5Q4S6cX/f+R4cZAhcvsPH6WfRpZ9TxYTq3FE2uk8cTxfxIF3kYjNwF7O
ma6YXQcCgYEApqdVVr92YSYZuDTKYGN5Rd5BaJSGPYWpShS5lPKBQ4n/TkSZzO1v
Bxq2AbPRohXasWEAdfgfss1mcwX9xnKO/DhnxXE8KVFXKVehGmUth0WqA5qtL4xD
h4hm3V7DcMqYbVzpmlWXFz5Is1fIflAhlfeX4h3s1rirTj+T6hODVf4=
-----END RSA PRIVATE KEY-----</pvt>
     <cert>-----BEGIN CERTIFICATE-----
MIIDiTCCAnECBETUDecwDQYJKoZIhvcNAQEEBQAwgYgxCzAJBgNVBAYTAlJVMQww
CgYDVQQIEwNOU08xDDAKBgNVBAcTA05TSzEUMBIGA1UEChMLRGFtYWdlIEluYy4x
CzAJBgNVBAsTAlFBMRcwFQYDVQQDEw53d3cuc3dzb2Z0LmNvbTEhMB8GCSqGSIb3
DQEJARYScmJ1c3lndWluQHBsZXNrLnJ1MB4XDTA2MDgwNTAzMTgwMVoXDTA3MDgw
NTAzMTgwMVowgYgxCzAJBgNVBAYTAlJVMQwwCgYDVQQIEwNOU08xDDAKBgNVBAcT
A05TSzEUMBIGA1UEChMLRGFtYWdlIEluYy4xCzAJBgNVBAsTAlFBMRcwFQYDVQQD
Ew53d3cuc3dzb2Z0LmNvbTEhMB8GCSqGSIb3DQEJARYScmJ1c3lndWluQHBsZXNr
LnJ1MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyxfczy4HqgwOI6yx
Ea+dLaMLe5zC0ijcawsha8oFtJlxv/DM9INpdKjv9HaInrR0StjG9HgqTpYrhOCJ
xeB07/gGsQ82xsvbcKANuCSQKInpwVimGUsisGNrfnbKIoAewN21ENCZQrufcQNW
NzTx/GjVCXqa/Sy4pXJ6Jud0dzA/+VisSBw5Oo4IqI2MDvuoDk+exQLdgM45Vgq4
d3A9+ESzmRmjmdovhNxnXdvMxA5RSrF+GZlBTFftsLQsnfnNiMR6ZTFgNCt2vdDm
UAV9iix2B1PuoH94TRCJRG5tVCFtfVzSRjo4ZzoiUs57efb3Oe56Lb64Ubi9ZRzN
hdkGLwIDAQABMA0GCSqGSIb3DQEBBAUAA4IBAQBH8mTX3Q3csBUjzwfy3kIVobgh
yElljOQ87Qv5rht5yktuTyS3oGXsDD0sO/uSG/akL34CTPkjl/vqYtzKMsfQ5pXY
MlY6Q+GCd9FgL5pBn1S8HSZLpTBWZc25mNe3mXbCQzI03r4W+dQajAgAgDKpnRjg
mblRg98+HwOL033pVgUnRwPoS3LO5jia5z3F0MkS8sV3x18DuoSLeVILhj0ttZ/p
B7x0kIUee8A95Q00EDh+4IaPSMOqiFrVIlsHEuPV33aCAz2Dk2TxzplsoNz61BFA
i3Cm04Gz1h9W/yzkcYCqiwUMIzSgUSBLn0hBeTid1u/NaDtic776YGuyaI+/
-----END CERTIFICATE-----</cert>
```

<ca>

```
-----BEGIN CERTIFICATE-----

MIIEjDCCA3SgAwIBAgIJAL4AGzKhkL5wMA0GCSqGSIb3DQEBBQUAMIGKMQswCQYD

VQQGEwJSVTELMAkGA1UECBMCTlYxFDASBgNVBAcTC05vdm9zaWJpcnNrMRIwEAYD

VQQKEwlQYXJhbGxlbHMxDjAMBgNVBAsTBVBsZXNrMQ0wCwYDVQQDEwRhYmVsMSUw

IwYJKoZIhvcNAQkBFhZhYmVseWFldkBwYXJhbGxlbHMuY29tMB4XDTA5MDQyODAz

NDMxMloXDTEwMDQyODAzNDMxMlowgYoxCzAJBgNVBAYTAlJVMQswCQYDVQQIEwJO

VjEUMBIGA1UEBxMLTm92b3NpYmlyc2sxEjAQBgNVBAoTCVBhcmFsbGVsczEOMAwG

A1UECxMFUGxlc2sxDTALBgNVBAMTBGFiZWwxJTAjBgkqhkiG9w0BCQEWFmFiZWx5

YWV2QHBhcmFsbGVscy5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB

AQDlslx5nXTQTvoic3GgW4HH5n5PoCz95Z9XNtCykwv0M2HdhMeNvde1NYDvkipa

FHzVPEH7eBgjAsHR4y1J4zVGAs2KiJF0W81vi5YAVMOJ7Ysyz8WLP0PQWTmUoMFM

aE0P7m788tphL328chdPJTjlCF6w9FKzddzrvdeK04ojp2cRO6fFMH7WJPcajvh/
```

3EuYpc7xBNC6Wf8Gk6tB5kCCe7wpHyXsc7ve97nn30p6rUXypBNHYmLujMCtS90I

K2xOCaSsCwkeUQ4mpXrr5lK7yM07b322vNZBiMTHV8DnaOWDPIcEIFl8NmOJovub

dG5wdXopA6seCQLaf95lvULrAgMBAAGjgfIwge8wHQYDVR0OBBYEFLSg4V0zc0i0

pRFtcygaiTgz5ikCMIG/BgNVHSMEgbcwgbSAFLSg4V0zc0i0pRFtcygaiTgz5ikC

oYGQpIGNMIGKMQswCQYDVQQGEwJSVTELMAkGA1UECBMCTlYxFDASBgNVBAcTC0
5v

dm9zaWJpcnNrMRIwEAYDVQQKEwlQYXJhbGxlbHMxDjAMBgNVBAsTBVBsZXNrMQ0w

CwYDVQQDEwRhYmVsMSUwIwYJKoZIhvcNAQkBFhZhYmVseWFldkBwYXJhbGxlbHMu

Y29tggkAvgAbMqGQvnAwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQUFAAOCAQEA

dEkV4DZMr7Rp0dlYqDpdviGu7FE+7R5u87Mz3E9cTApycbg4KTZAS53B3ItsAQv2

yjXnkTQw2/l5XusW+OxhGpWC0z58jScyqNW/HkvheUrgbylIOOaKs9vAayi3mA0T

O1cB1vnAgmB8puV6IdEKuvUimgbJhiGIpqTiL+2/YgSnkGhRuYlxuFB6U8WPgrUN

UMt0linQtUsxMJCghMA3T8cODGon4ugZ0cCJmDfpkStI3jUuoxulSX2I8xmUQtGe

2Q/sUQaWgJhe4RDoth6w7E9GE8733WGhC1mIjxyJMx4vmu4ofLAwN6XUzJJ3eC7s

P34vOGq/sqxG/U+BbUu71A==

-----END CERTIFICATE-----

</ca>

</content>

<ip_address>192.0.2.14</ip_address>

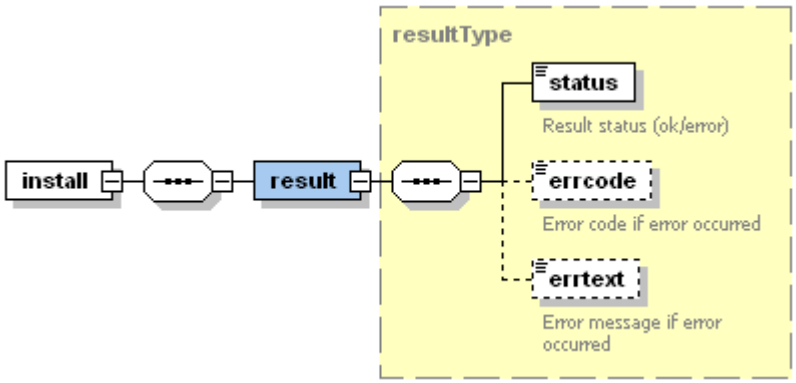
</install>

</certificate>

</packet>

Struktur des Antwortpakets

Der **install**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort.
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string.* Zulässige Werte: ok | error.
- **errcode**, *erforderlich*, falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *unsignedInt.*
- **errtext**, *erforderlich*, falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string.*

Antwort-Beispiele

Die positive Antwort von dem Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<certificate>
<install>
  <result>
    <status>ok</status>
  </result>
</install>
</certificate>
</packet>
```

Negative Antworten vom Server können wie folgt aussehen:

```
<packet version=”1.6.2.0”>
<certificate>
<install>
  <result>
    <status>error</status>
    <errcode>8002</errcode>
  <errtext>Unable to set csr content to certificate : CSR field contains
an improper value : openssl failed: </errtext>
  </result>
</install>
</certificate>
</packet>
```

```
<packet version=”1.6.2.0”>
<certificate>
<install>
  <result>
    <status>error</status>
    <errcode>8006</errcode>
    <errtext>Unable to set certificate name : </errtext>
  </result>
</install>
```

</certificate>

</packet>

Erstellen von Domains

Für die Erstellung eines Domain-Accounts erstellen Sie ein XML-Paket mit dem **domain/add**-Kontrollknoten:

<packet version=”1.6.2.0”>

<domain>

```
  <add>
    …
  </add>
```

</domain>

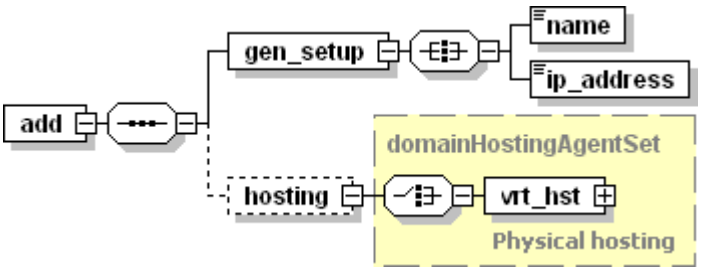
</packet>

In diesem Abschnitt:

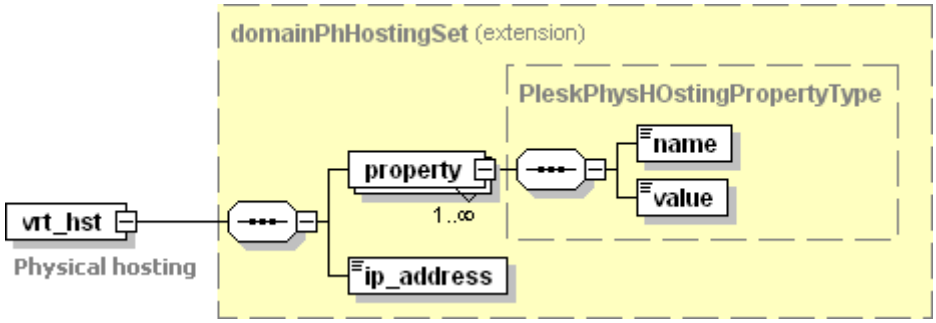
Struktur des Abfrage-Pakets.....135
Beispiele für Abfragen.....137
Struktur des Antwortpakets.....138
Antwort-Beispiele.....139

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket, über das ein Domain-Account erstellt wird, beinhaltet den **del_rec**-Kontrollknoten, der wie folgt strukturiert ist:



- **gen_setup**, *erforderlich*
  Beinhaltet die wichtigsten Informationen über den Domain-Account. Datentyp: *none*.
  - **name**, *erforderlich*
    Gibt den Domainnamen an. Datentyp: *domainName* (`plesk_domain.xsd`).
  - **ip_address**, *erforderlich*
    Gibt die IP an, auf der die Domain gehostet wird. Datentyp: *ip_address* (`common.xsd`).
- **hosting**, *erforderlich*
  Gibt die Hostingeinstellungen für die Domain an. Datentyp: *domainHostingAgentSet* (`plesk_domain.xsd`).
- **vrt_hst**, *erforderlich*
  Gibt die Hostingeinstellungen für die Domain an. Datentyp: *domainPhHostingSet* (`plesk_domain.xsd`).
  Wie folgt strukturiert:



- **property**, *erforderlich*
  Gibt einen Hosting-Parameter an. Datentyp: *PleskPhysHostingPropertyType* (`plesk_domain.xsd`).
  - **name**, *erforderlich*
    Gibt einen Hosting-Parameter-Namen an. Datentyp: *string*.
  - **value**, *erforderlich*
    Gibt einen Hosting-Parameter-Wert an. Datentyp: *any*.
- **ip_address**, *erforderlich*
  Gibt die IP-Adresse der Domain an. Datentyp: *ip_address* (`common.xsd`).

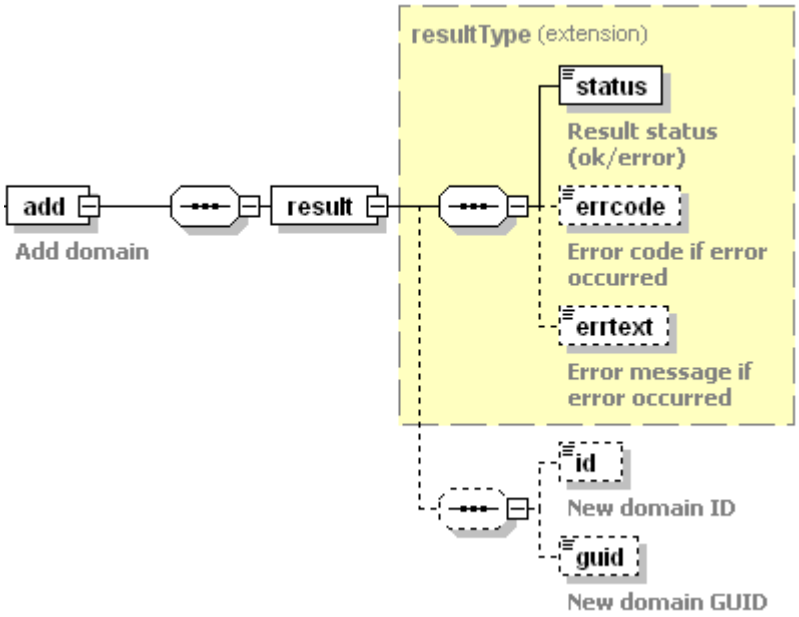
Beispiele für Abfragen

Diese Abfrage erstellt die Domain example.com, die auf der IP 192.0.2.48 gehostet wird.

```
<packet version=”1.6.2.0”>
<domain>
    <add>
    <gen_setup>
        <name>example.com</name>
        <ip_address>192.0.2.48</ip_address>
    </gen_setup>
    <hosting>
        <vrt_hst>
        <property>
            <name>ftp_login</name>
            <value>fp16se4fdf0</value>
        </property>
        <property>
            <name>ftp_password</name>
            <value>qweqwe</value>
        </property>
        <ip_address>192.0.2.48</ip_address>
        </vrt_hst>
    </hosting>
    </add>
</domain>
</packet>
```

Struktur des Antwortpakets

Der **add**-Knoten des Antwortpakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **id**, *erforderlich, falls die Operation erfolgreich abgeschlossen wird*
  Liefert den eindeutigen Identifier des gerade erstellten Domain-Accounts. Datentyp: *integer*.
- **guid**, *erforderlich , falls die Operation erfolgreich abgeschlossen wird*
  Liefert den GUID (Global Unique Identifier bezeichnet eine global eindeutige Zahl) des gerade erstellten Domain-Accounts. Datentyp: *string.*

Antwort-Beispiele

Eine positive Antwort, die vom Server nach dem Hinzufügen eines neuen Domain-Accounts empfangen wird, sieht wie folgt aus:

<packet version=”1.6.2.0”>

<domain>

<add>

<result>

<status>ok</status>

<id>6</id>

<guid>5c0e3881-22a2-4401-bcc0-881d691bfdef</guid>

</result>

</add>

</domain>

</packet>

Eine negative Antwort kann wie folgt aussehen (eventuell wird bei Ihnen ein anderer Fehlercode angezeigt, je nachdem wodurch der Fehler verursacht wurde):

<packet version=”1.6.2.0”>

<domain>

<add>

<result>

<status>error</status>

<errcode>1023</errcode>

<errtext>Operation failed.</errtext>

</result>

</add>

</domain>

</packet>

Folgende negative Antwort wird empfangen, wenn ein Abfrage-Paket nicht alle erforderlichen Hostingeinstellungen angibt:

ABFRAGE-PAKET

<packet version=”1.6.2.0”>

```
<domain>
  <add>
    <gen_setup>
    <name>sample.com</name>
    <ip_address>10.53.129.101</ip_address>
    </gen_setup>
    <hosting>
       <vrt_hst>
    <property>
         <name>ftp_password</name>
         <value>fp16sdfdfrttg0</value>
    </property>
    <ip_address>10.53.129.101</ip_address>
    </vrt_hst>
    </hosting>
  </add>
  </domain>
```

</packet>

ANTWORTPAKET

<packet version="1.6.2.0">

```
  <domain>
```

<add>

```
    <result>
    <status>error</status>
    <errcode>2204</errcode>
  <errtext>Unable to update hosting preferences. system user update is
failed: Unable to check system user existence: login name is empty.
Incorrect fields: "login".</errtext>
    </result>
    </add>
  </domain>
```

</packet>

Erstellen von Subdomains

Für die Erstellung eines Subdomain-Accounts erstellen Sie ein XML-Paket mit dem **subdomain/add**-Kontrollknoten:

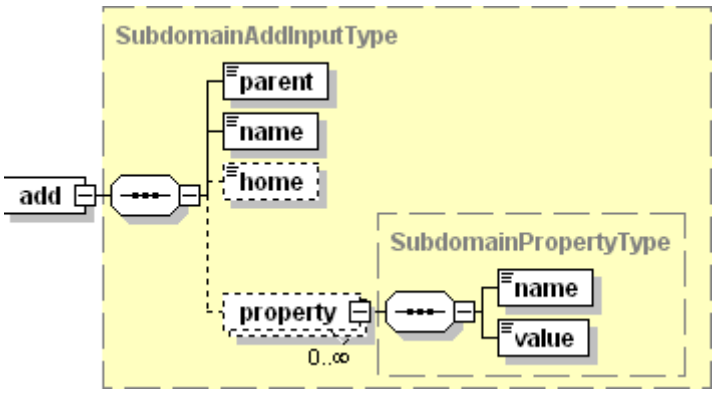
```
<packet version="1.6.2.0">
<subdomain>
  <add>
    …
  </add>
</subdomain>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets..... 141
Beispiele für Abfragen..... 143
Struktur des Antwortpakets..... 144
Antwort-Beispiele..... 145

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket, über das eine neue Subdomain hinzugefügt wird, beinhaltet den **subdomain/add**-Kontrollknoten, der wie folgt strukturiert ist:



- **parent**, *erforderlich*

  Gibt den Namen der Domain/Subdomain an, auf der eine Subdomain erstellt wird. (Übergeordnete (“Parent”) Subdomains werden nur auf Windows unterstützt.) Datentyp: *string*.

- **name**, *erforderlich*

  Gibt den Namen der Subdomain an. Datentyp: *string*.

- **home**, *erforderlich*, falls eine Subdomain auf einem Unterordner erstellt wird (nur bei Windows)

  Gibt den Pfad zum Home-Verzeichnis der Subdomain an. Datentyp: *string*.

  Falls es hier keinen Eintrag gibt (`<home/>`), wird das root-Verzeichnis der übergeordneten (“Parent”) Domain verwendet.

- **property**, *optional*

  Definiert eine Hostingeinstellung der erstellten Subdomain. Datentyp: *SubdomainPropertyType* (`subdomain.xsd`).

  - **name**, *erforderlich*

    Gibt einen Hosting-Parameter-Namen an. Datentyp: *string*.

  - **value**, *erforderlich*

    Gibt einen Hosting-Parameter-Wert an. Datentyp: *any*.

Beispiele für Abfragen

Dieses Paket erstellt die Subdomain *forum.example.com*:

```
<packet version=”1.6.2.0”>
<subdomain>
<add>
    <parent>example.com</parent>
    <name>forum</name>
    <property>
    <name>ftp_login</name>
    <value>john</value>
    </property>
    <property>
    <name>ftp_password</name>
    <value>sample</value>
    </property>
</add>
</subdomain>
</packet>
```

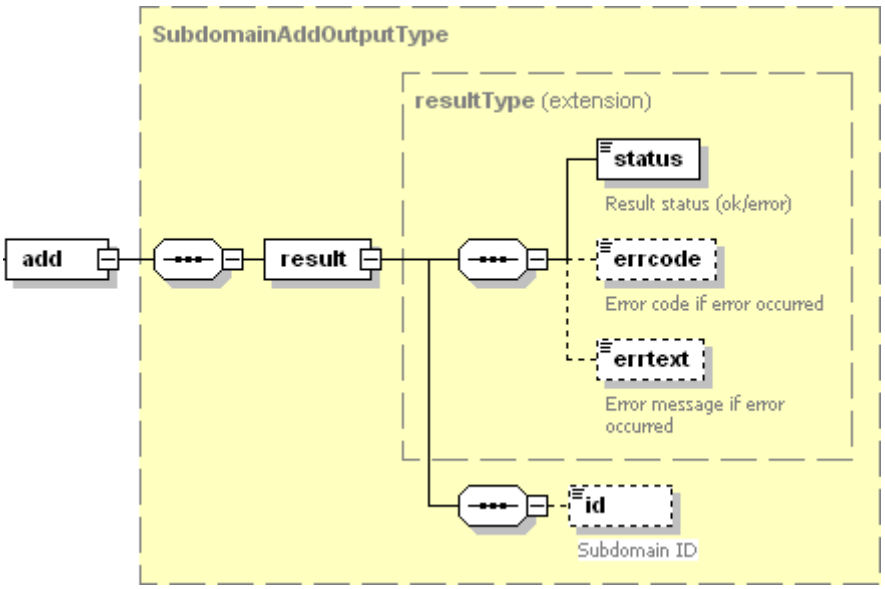
Eine Subdomain auf einem Unterordner erstellen

Dieses Paket erstellt eine Subdomain *blog.example.com* auf dem Unterordner */httpdocs/BlogEngine*:

```
<packet version=”1.6.2.0”>
<subdomain>
<add>
    <parent>example.com</parent>
    <name>blog</name>
    <home>/httpdocs/BlogEngine</home>
</add>
</subdomain>
</packet>
```

Struktur des Antwortpakets

Der **add**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **id**, *erforderlich, falls die Operation erfolgreich abgeschlossen wird*
- Liefert den eindeutigen Identifier erstellten Subdomain. Datentyp: *id_type* (`common.xsd`).

Operations-spezifische Fehler

- 1007 - Subdomain mit einem solchen Namen existiert bereits.
- 1015 - Übergeordnete “Parent”-Domain/Subdomain wurde nicht gefunden.
- 1019 - Ungültige Eigenschaft angegeben.
- 1023 - Operation fehlgeschlagen.

Antwort-Beispiele

Eine positive Antwort vom Server nach der Erstellung der Subdomain sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<subdomain>
    <add>
    <result>
        <status>ok</status>
        <id>1</id>
    </result>
    </add>
</subdomain>
</packet>
```

Folgende negative Antwort wird empfangen, falls die angegebene Subdomain bereits existiert:

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<subdomain>
    <add>
    <parent>example.com</parent>
    <name>blog</name>
    </add>
</subdomain>
</packet>
```

ANTWORTPAKET

```
<packet version=”1.6.2.0”>
<subdomain>
    <add>
    <result>
        <status>error</status>
        <errcode>1007</errcode>
     <errtext>The subdomain with such name already exists.</errtext>
    </result>
    </add>
```

```
</subdomain>
</packet>
```

Verfügbare APS-Kataloge definieren

Um eine benutzerdefinierte Liste (Konfigurationsdatei) des APS-Katalogs in das Panel zu importieren, erstellen Sie ein XML-Abfrage-Paket mit dem **aps/import-config**-Kontrollknoten:

```
<packet version="1.6.2.0">
<aps>
  <import-config>
...
  </import-config>
</aps>
</packet>
```

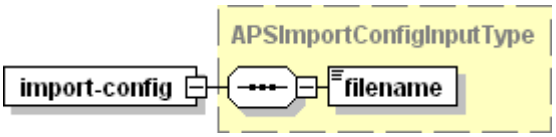
Die Konfigurationsdatei (auf Seite 49) sollte zuvor auf den Server mithilfe des **upload**-Operators (auf Seite 151) hochgeladen werden.

In diesem Abschnitt:

Struktur des Abfrage-Pakets 146
Beispiele für Abfragen 147
Struktur des Antwortpakets 148
Antwort-Beispiele 149

Struktur des Abfrage-Pakets

Der **import-config**-Knoten des XML-Abfrage-Pakets ist wie folgt strukturiert:



- **filename**, *erforderlich*

  Gibt den Namen der Konfigurationsdatei an, die auf den Server hochgeladen wird. Datentyp: *string*.

  Der Wert sollte dem Namen der temporären Datei entsprechen, die in der **upload**-Antwort (siehe Seite 160) (in dem `"upload/result/file"`-Element) wiedergegeben wird.

Beispiele für Abfragen

Abfrage-Pakete, die Konfigurationsdateien von APS-Katalogen importieren, die zuvor auf den Server hochgeladen wurden, sehen wie folgt aus.

Auf Linux/Unix:

<packet version="1.6.2.0">

<aps>

<import-config>

<filename>/usr/local/psa/tmp/li_8FZruf</filename>

</import-config>

</aps>

</packet>

Auf Windows:

<packet version="1.6.2.0">

<aps>

<import-config>

<filename>"C:/Program Files/Parallels/Plesk/tmp/li_9D.tmp"</filename>

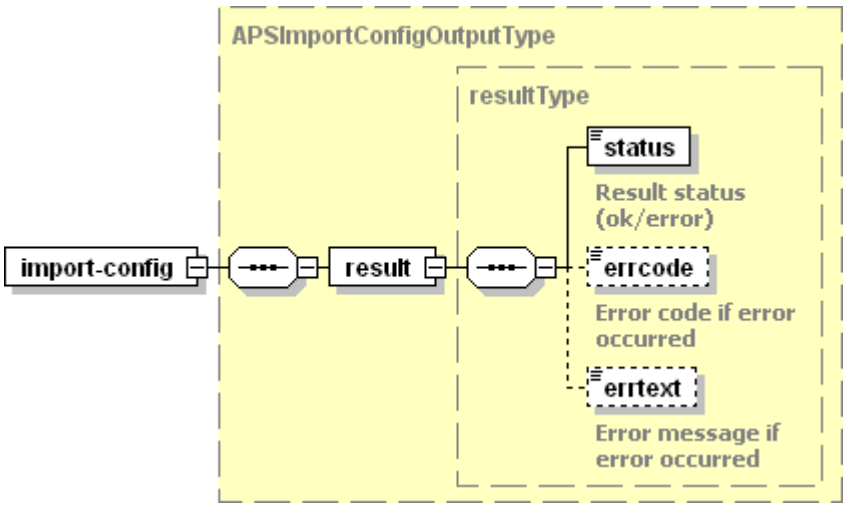
</import-config>

</aps>

</packet>

Struktur des Antwortpakets

Der **import-config**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version="1.6.2.0">
<aps>
    <install>
    <result>
      <status>ok</status>
    </result>
    </install>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, falls keine Abfrage-Datei gefunden werden konnte:

```
<packet version="1.6.2.0">
<aps>
    <import-config>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
      <errtext>File "/usr/local/psa/tmp/li_8FZruf" does not
   exist</errtext>
    </result>
    </import-config>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, falls eine Abfrage-Datei über unzulässige Zugangsberechtigungen verfügt (höchstwahrscheinlich, wenn für das Hochladen eine andere Methode als der **upload**-Agent verwendet wird):

```
<packet version="1.6.2.0">
<aps>
    <import-config>
    <result>
        <status>error</status>
```

```
      <errcode>1023</errcode>
    <errtext>Unable to use specified file as config file</errtext>
  </result>
  </import-config>
</aps>
</packet>
```

Installieren von Anwendungen

Um Anwendungen auf Domains und Subdomains zu installieren, verwenden Sie den **aps**-Operator wie im Folgenden beschrieben.

In diesem Abschnitt:

Import von Anwendungspaketen 150
Installieren einer Anwendung 185

Import von Anwendungspaketen

In diesem Abschnitt:

Dateien auf den Server hochladen 151
Import von hochgeladenen Anwendungspaketen 163
Anwendungspakete vom APS-Katalog herunterladen 168
Status eines Download-Tasks abfragen 174

Dateien auf den Server hochladen

Um Dateien auf den vom Panel verwalteten Server hochzuladen, verwenden Sie den **upload**-Operator. Im Gegensatz zu anderen Operatoren, verwendet der **upload**-Operator keine Standardstruktur des Abfrage-Pakets. Es verwendet stattdessen die POST-Methode (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html), um Dateien hochzuladen.

Wenn eine Abfrage von dem Panel verarbeitet wird, wird die Antwort im regulären XML-Antwortpaket wiedergegeben.Nachdem eine Datei auf das Panel hochgeladen wurde, kann sie für andere Prozesse verwendet werden.

Hinweis: Sie können Dateien auch über den FTP-Manager hochladen, aber beachten Sie sich in einem solchen Fall bitte, dass die Berechtigungen zu den hochgeladenen Dateien, unzulässig für weitere API RPC-Operationen auf den Dateien sind. Aus diesem Grund empfehlen wir die Verwendung des **upload**-Operators.

In diesem Abschnitt:

Dateien mit cURL hochladen 151
Dateien mit PHP hochladen 153
Dateien mit .NET hochladen 155
Struktur des Antwortpakets 160
Antwort-Beispiele 161

Dateien mit cURL hochladen

Um eine Datei hochzuladen, befolgen Sie bitte die folgenden Schritte:

1. Erstellen Sie eine SSH_Verbindung zu dem Unix-Server, auf dem sich die Datei befindet.
2. Geben Sie den folgenden String in die Unix-Shell ein:

```
curl -k -F myfile=@install.log -H 'HTTP_AUTH_LOGIN:admin' -H
'HTTP_AUTH_PASSWD:password' https://panel-ip:8443/enterprise/control/agent.
php
```

Argumente

curl

Legt fest, dass cURL (http://curl.haxx.se/) verwendet wird. Sie müssen diesen Client auf den Unix-Maschine installieren.

-k

Die SSL-Verbindung wird verwendet.

-F myfile=@install.log

Die Datei `install.log` soll hochgeladen werden. Sie können mehrere Dateien in einem einzigen Paket hochladen.

-H 'HTTP_AUTH_LOGIN:admin'

Gibt den Login des Panel-Administrators an. Ersetzen Sie den Admin-Wert durch Ihren Panel-Login.

-H 'HTTP_AUTH_PASSWD:password'

Gibt das Passwort des Panel-Administrators an. Ersetzen sied en Passwort-Wert durch Ihre Passwort für den Zugang zum Panel.

https://panel-ip:8443/enterprise/control/agent.php

Gibt die Adresse des Panel-Servers an. Ersetzen Sie den Wert *<panel-ip>* durch die IP-Adresse des Servers.

3. Drücken Sie die EINGABETASTE.

Dateien mit PHP hochladen

Unten wird ein Beispiel für ein PHP-Skript angegeben, um Dateien auf das Panel hochzuladen. Ersetzen Sie HOST, LOGIN, PASSWD und FILENAME durch die Panel-Anmeldeinformationen.

HOST	Die IP-Adresse oder der Name des vom Panel verwalteten Servers.
Login	Login-Name des Panel-Administrators.
PASSWD	Passwort des Panel-Administrators.
FILENAME	Vollständiger Name der Datei, die hochgeladen wird.

```
<?php
define(“HOST”, “10.58.97.31”);
define(“PATH”, “/enterprise/control/agent.php”);
define(“LOGIN”, “admin”);
define(“PASSWD”, ‘setup’);
define(“FILENAME”, ‘../../data.rpm’);

function write_callback($ch, $data)
{
echo $data;
return strlen($data);
}

function uploadFile($filename)
{$url = “https://” . HOST . “:8443” . PATH;
$headers = array(
“HTTP_AUTH_LOGIN: “ . LOGIN,
“HTTP_AUTH_PASSWD: “ . PASSWD,
“HTTP_PRETTY_PRINT: TRUE”,
“Content-Type: multipart/form-data;”,
);

// Initialize the curl engine
$ch = curl_init();

// Set the curl options
```

```
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
// this line makes it work under https
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
// Set the URL to be processed
curl_setopt($ch, CURLOPT_URL, $url);

curl_setopt($ch, CURLOPT_POSTFIELDS, array
('sampfile'=>"@$filename"));

$result = curl_exec($ch);
if (curl_errno($ch)) {
echo "\n\n-------------------------\n" .
"cURL error number:" .
curl_errno($ch);
echo "\n\ncURL error:" . curl_error($ch);
}

curl_close($ch);
//fclose($fp);
return;
}

uploadFile(realpath(FILENAME));

?>
```

Dateien mit .NET hochladen

Diese .NET-Quellcode lädt Dateien auf das Panel hoch. Ersetzen Sie die Werte der Variablen `Hostname`, `Login`, `Password` und `Protocol` durch die Parameter der Panel-Instanz. Verändern Sie den Wert der Variablen `Filename` in den vollständigen Namen der Datei, die hochgeladen werden soll.

Filename	Vollständiger Name der Datei, die hochgeladen werden soll.
Hostname	Die IP-Adresse oder der Name des vom Panel verwalteten Servers.
Login	Login-Name des Panel-Administrators.
Password	Passwort des Panel-Administrators.
Protocol	Version des API RPC-Protokolls.

```
using System;
using System.Net;
using System.Text;
using System.IO;
using System.Xml;
using System.Xml.Schema;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
namespace ConsoleApplication1
{
public class Request
 {
  // Public interface
  //
  public string Filename = "./tmp/myfile.sh";   // Upload file name;
  public string Hostname = "localhost";   // The Panel's host name
  public string Login = "admin";          // Administrator Login
  public string Password = "setup";       // Administrator Password
  public string Protocol = "1.6.2.0";     // API RPC Version Protocol.
  // Handler for receiving information about document type definition
(DTD),
  // XML-Data Reduced (XDR) schema, and XML Schema definition language
(XSD) schema validation errors.
  public ValidationEventHandler XmlSchemaValidation = null;
  public Request()
```

```
  {
  }

  public string AgentEntryPoint { get { return "https://" + Hostname +
":8443/enterprise/control/agent.php"; } }

  public string InputValidationSchema { get { return "https://" +
Hostname + ":8443/schemas/rpc/" + Protocol + "/agent_input.xsd"; } }

  public string OutputValidationSchema { get { return "https://" +
Hostname + ":8443/schemas/rpc/" + Protocol + "/agent_output.xsd"; } }

  public XmlDocument UploadFile(string uploadfile)

  {
  HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(AgentEntryPoint);

  string boundary = "----------" + DateTime.Now.Ticks.ToString("x");

   request.Headers.Add("HTTP_AUTH_LOGIN",
   Login); request.Headers.Add("HTTP_AUTH_PASSWD",
   Password); request.ContentType = "multipart/form-data; boundary=" +
   boundary; request.Method = "POST";

    // Build up the post message header

    StringBuilder sb = new StringBuilder();

    sb.Append("—");

    sb.Append(boundary);

    sb.Append("\r\n");

  sb.Append("Content-Disposition: form-data; name=\"");

    sb.Append("sampfile");

    sb.Append("\"; filename=\"");

    sb.Append(Path.GetFileName(uploadfile));

    sb.Append("\"");

    sb.Append("\r\n");

    sb.Append("Content-Type: ");

    sb.Append("application/octet-stream");

    sb.Append("\r\n");

    sb.Append("\r\n");

   string postHeader = sb.ToString();

   byte[] postHeaderBytes = Encoding.UTF8.GetBytes(postHeader);

      // Build the trailing boundary string as a byte array
      // ensuring the boundary appears on a line by itself
      byte[] boundaryBytes =
Encoding.ASCII.GetBytes("\r\n—" + boundary + "\r\n");
   FileStream fileStream = new FileStream(uploadfile,

   FileMode.Open, FileAccess.Read);
```

```
  long length = postHeaderBytes.Length + fileStream.Length +
boundaryBytes.Length;
    request.ContentLength = length;
    Stream stream = request.GetRequestStream();
  stream.Write(postHeaderBytes, 0, postHeaderBytes.Length);
// Write out the file contents
  byte[] buffer = new Byte[checked((uint)Math.Min(4096,
(int)fileStream.Length))];
   int bytesRead = 0;
   while ((bytesRead = fileStream.Read(buffer, 0, buffer.Length)) != 0)
   stream.Write(buffer, 0, bytesRead);
   // Write out the trailing boundary stream.Write(boundaryBytes, 0,
   boundaryBytes.Length);
    XmlDocument result = GetResponse(request);
    return result;
  }
  // Private interface
  //
   // Parsing and validating packet
  //
  private XmlDocument ParseAndValidate(TextReader xml, string
schemaUri)
  {
    XmlSchemaSet schemas = new XmlSchemaSet();
    schemas.Add(null, schemaUri);
   XmlReaderSettings settings = new XmlReaderSettings();
   if (XmlSchemaValidation != null)
  settings.ValidationEventHandler += new
ValidationEventHandler(XmlSchemaValidation);
  settings.ValidationType = ValidationType.Schema;
  settings.ValidationFlags |=
XmlSchemaValidationFlags.ProcessSchemaLocation;
  settings.Schemas = schemas;
  XmlDocument document = new XmlDocument();
  using (XmlReader reader = XmlTextReader.Create(xml, settings))
   {
```

```
    document.Load(reader);
   }

  return document;
  }
  private XmlDocument GetResponse(HttpWebRequest request)
  {
  using (HttpWebResponse response =
  (HttpWebResponse)request.GetResponse())
  using (Stream stream = response.GetResponseStream())
  using (TextReader reader = new StreamReader(stream))
   {
    return ParseAndValidate(reader, OutputValidationSchema);
   }
  }
 }
```

class Program

```
 {
  static void Main(string[] args)
  {
  if (args.Length < 4)
   {
  Console.WriteLine("Usage: ParalLelSpaNelapirpcclient <Hostname>
<Login> <Password> <FileName>");
    Console.WriteLine(" ");
    Console.WriteLine(" Hostname - The Panel's host name");
    Console.WriteLine(" Login  - Administrator login");
    Console.WriteLine(" Password - Administrator password");
  Console.WriteLine(" FileName - Upload file name");
  return;
   }

  // Verifies the remote Secure Sockets Layer (SSL) certificate used for
authentication.
```

ServicePointManager.ServerCertificateValidationCallback =

```
    new
  RemoteCertificateValidationCallback(RemoteCertificateValidation);
  Request request = new Request(); request.XmlSchemaValidation =
  XmlSchemaValidation;
  try
   {
    XmlDocument result = request.UploadFile(filename);
```

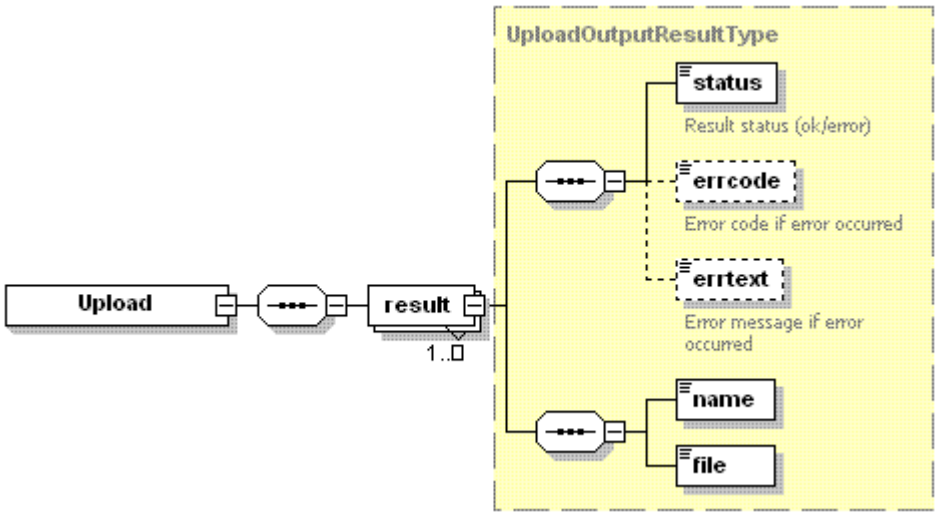
```
    PrintResult(result);
   }
  catch (Exception e)
   {
    Console.WriteLine("Request error: {0}", e.Message);
   }
  }

  // The following method is invoked by the
RemoteCertificateValidationDelegate.
  private static bool RemoteCertificateValidation(object sender,
  X509Certificate certificate, X509Chain chain, SslPolicyErrors
sslPolicyErrors)
  {
  if (sslPolicyErrors !=
SslPolicyErrors.RemoteCertificateNotAvailable)
    return true;
  Console.WriteLine("Certificate error: {0}", sslPolicyErrors);
   // Do not allow this client to communicate with unauthenticated
   servers.  return false;
  }
  //
  private static void XmlSchemaValidation(object sender,
ValidationEventArgs e)
  {
  Console.WriteLine("Validation error: {0}", e.Message);
  }

  static void PrintResult(XmlDocument document)
  {
   XmlTextWriter writer = new XmlTextWriter(Console.Out);
   writer.Formatting = Formatting.Indented;
   document.WriteTo(writer);
    writer.Flush();
    Console.WriteLine();
  }
 }
}
```

Struktur des Antwortpakets

Der **upload**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **name**, *erforderlich*
  Gibt den Namen der hochgeladenen Datei an. Datentyp: *string*.
- **file**, *erforderlich*
  Gibt den vollständigen Namen der erstellten, temporären Datei an. Datentyp: *string*.

Antwort-Beispiele

Hochladen eines einzigen Pakets

Wenn ein Agent anfragt, das `li.sh`-Paket auf den vom Panel verwalteten Server hochzuladen, sieht eine positive Antwort von dem Server wie folgt aus:

<packet version="1.6.2.0">

```
  <upload>
```

<result>

```
    <status>ok</status>
    <name>li.sh</name>
    <file>/usr/local/psa/tmp/li_8FZruf</file>
     </result>
  </upload>
</packet>
```

Falls die hochgeladene Datei die zulässige Dateigröße *upload_max_filesize* aus `php.ini` überschreitet, sieht die Antwort vom Server wie folgt aus:

<packet version="1.6.2.0">

```
  <upload>
```

<result>

```
    <status>error</status>
    <errcode>1023</errcode>
  <errtext>The uploaded file exceeds the upload_max_filesize directive
set in php.ini</errtext>
```

<name>li.sh</name>

```
    <file>/usr/local/psa/tmp/li_8FZruf</file>
    </result>
  </upload>
</packet>
```

Hochladen von mehreren Paketen

Wenn ein Agent die Anfrage stellt, die Dateien `li.sh` und `mybox.sh` auf den Panel-Server hochzuladen, dann sieht eine positive Antwort von dem Server wie folgt aus:

<packet version="1.6.2.0">

```
<upload>
```

<result>

```
<status>ok</status>
<name>li.sh</name>
<file>/usr/local/psa/tmp/li_8FZruf</file>
</result>
<result>
<status>ok</status>
<name>mybox.sh</name>
<file>/usr/local/psa/tmp/li_8FZrHts</file>
</result>
</upload>
</packet>
```

Import von hochgeladenen Anwendungspaketen

Um ein hochgeladenes Anwendungspaket (auf Seite 151) in das Panel zu importieren, erstellen Sie ein XML-Abfrage-Paket mit dem **aps/import-package**-Kontrollknoten:

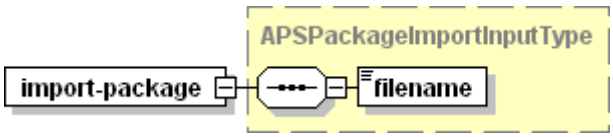
```
<packet version=”1.6.2.0”>
<aps>
  <import-package>
...
  </import-package>
</aps>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets.....163
Beispiele für Abfragen.....164
Struktur des Antwortpakets.....165
Antwort-Beispiele.....166

Struktur des Abfrage-Pakets

Ein XML-Abfrage-Paket für den Import von APS-Paketen in das Panel enthält den **import-package**-Kontrollknoten und ist wie folgt strukturiert:



- **filename**, *erforderlich*

  Gibt den Namen des Konfigurationspakets an, die auf den Server hochgeladen wird. Datentyp: *string*.

  Der Wert sollte dem Namen der temporären Datei entsprechen, die in der **upload**-Antwort (siehe Seite 160) (in dem `“upload/result/file”`-Element) wiedergegeben wird.

Beispiele für Abfragen

Abfrage-Pakete, die Konfigurationsdateien von APS-Paketen in das Panel importieren, die zuvor auf den Server hochgeladen wurden, sehen wie folgt aus.

Auf Linux/Unix:

<packet version=”1.6.2.0”>

<aps>

<import-package>

<filename>/usr/local/psa/tmp/li_8FZruf</filename>

</import-package>

</aps>

</packet>

Auf Windows:

<packet version=”1.6.2.0”>

<aps>

<import-package>

<filename>”C:/Program Files/Parallels/Plesk/tmp/li_9D.tmp”</filename>

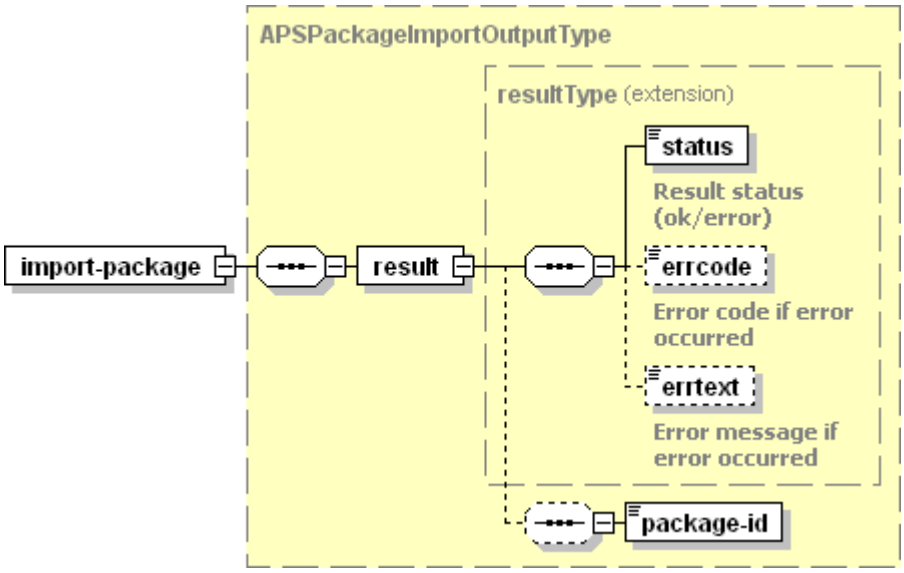
</import-package>

</aps>

</packet>

Struktur des Antwortpakets

Der **import-package**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **package-id**, *erforderlich*, falls die Operation erfolgreich abgeschlossen wird
  Gibt die ID an, die dem Paket auf dem vom Panel verwalteten Server zugewiesen wurde. Datentyp: *id_type* (`common.xsd`).

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<aps>
    <import-package>
    <result>
        <status>ok</status>
        <package-id>13</package-id>
    </result>
    </import-package>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, falls keine Abfrage-Datei gefunden werden konnte:

```
<packet version=”1.6.2.0”>
<aps>
    <import-package>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>File “/usr/local/psa/tmp/li_8FZruf” does not
  exist</errtext>
    </result>
    </import-package>
</aps>
</packet>
```

Folgende negative Antworten werden empfangen, falls eine importierte Datei kein gültiges APS-Anwendungspaket enthält.

1) Das zu importierende Archiv enthält keine `APP-META.xml`-Paket-Metadaten.

```
<packet version=”1.6.2.0”>
<aps>
    <import-package>
    <result>
```

```
      <status>error</status>
      <errcode>1023</errcode>
    <errtext>Entry 'APP-META.xml' not found</errtext>
  </result>
  </import-package>
</aps>
</packet>
```

2) `APP-META.xml` kann nicht über das XML-Schema validiert werden.

```
<packet version="1.6.2.0">
<aps>
  <import-package>
  <result>
    <status>error</status>
    <errcode>1023</errcode>
  <errtext>Unable to acquire metadata description</errtext>
  </result>
  </import-package>
</aps>
</packet>
```

Anwendungspakete vom APS-Katalog herunterladen

Um ein Anwendungspaket vom APS-Katalog herunterzuladen, erstellen Sie ein XML-Abfrage-Paket mit dem **aps/download-package**-Kontrollknoten:

```
<packet version="1.6.2.0">
<aps>
  <download-package>
...
  </download-package>
</aps>
</packet>
```

Falls die Operation erfolgreich durchgeführt wird, wird ein Download-Task erstellt.

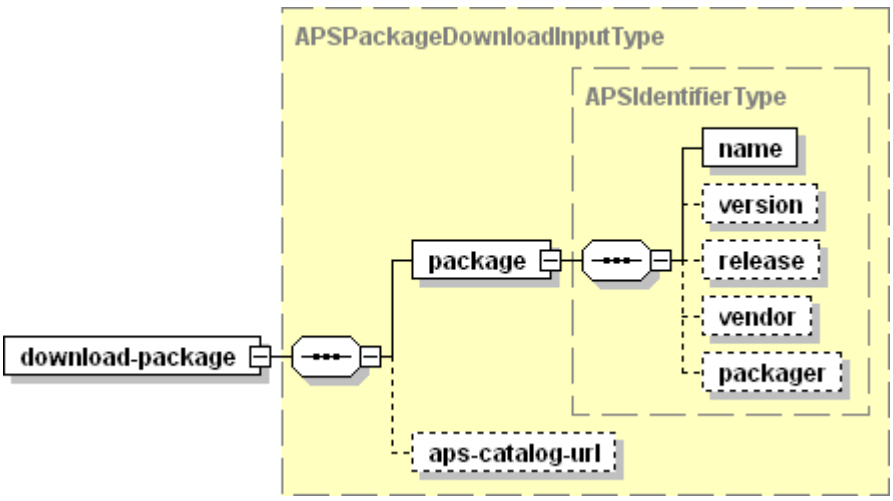
Hinweis: Eine Operation ist nicht nur dann erfolgreich, wenn ein Download-Vorgang gestartet wird. Ob ein Download-Vorgang tatsächlich erfolgreich war oder nicht, sollte mithilfe der **get-download-status** (auf Seite 174)-Operation überprüft werden.

In diesem Abschnitt:

Struktur des Abfrage-Pakets 168
Beispiele für Abfragen 170
Struktur des Antwortpakets 171
Antwort-Beispiele 172

Struktur des Abfrage-Pakets

Der **download-package**-Knoten ist wie folgt strukturiert:



- **package**, *erforderlich*

  Beinhaltet eine Reihe von Daten, die das Zielpaket beschreiben. Datentyp: *APSIdentifierType* (`aps.xsd`).

  Die Werte für die untergeordneten (“Child”) Knoten sollten den entsprechenden Elementen eines APS-Katalog Feed-Eintrags entnommen werden, der das zu herunterladene Paket beschreibt. Weitere detaillierte Informationen finden Sie im Abschnitt *Feeds: Referenz zu den Elementen* in der *API-Referenz des APS-Katalogs* (http://www.apsstandard.org/r/doc/aps-catalog-1.1-api/index.htm).

- **name**, *erforderlich*

  Gibt den Namen des Anwendungspakets an. Datentyp: *string*.

- **version**, *optional*

  Gibt die Version der Anwendung an. Datentyp: *string*.

  Falls nicht näher definiert und falls mehrere Versionen der Anwendung in dem APS-Katalog existieren, wird das Paket mit der neuesten Version heruntergeladen.

- **release**, *optional*

  Gibt die Paket-Release an. Datentyp: *string*.

  Falls nicht näher definiert und falls mehrere Releases der Anwendung in dem APS-Katalog existieren, wird das Paket mit der neuesten Version heruntergeladen.

- **vendor**, *optional*

  Beinhaltet Informationen über den Anbieter der Anwendung. Datentyp: *string*.

- **packager**, *optional*

  Beinhaltet Informationen über den Packager (“Verpacker”) der Anwendung. Datentyp: *string*

  aps-catalog-url, *optional*

  Beinhaltet die URL des APS-Katalogs von dem das Paket heruntergeladen werden soll. Datentyp: *string*

  Falls ein Knoten vorhanden ist, sollte URL in ihrem Wert exakt mit der aus der Konfigurationsdatei des APS-Katalogs (auf Seite 49) übereinstimmen, andernfalls erhalten Sie eine Fehlermeldung mit dem Code 1013.

  Falls nicht in einem Paket enthalten, dann wird das Paket von dem APS-Katalog heruntergeladen, das an erster Stelle in der Konfigurationsdatei des APS-Katalogs (auf Seite 146) definiert wird.

Beispiele für Abfragen

Dieses Paket installiert ein Anwendungspaket einer bestimmten Version, Release, Anbieter und Publisher und von einem ganz bestimmten APS-Katalog (höchstwahrscheinlich ein anderer als der standardmäßig voreingestellte).

<packet version=”1.6.2.0”>

<aps>

<download-package>

<package>

<name>BlogEngine.NET</name>

<version>1.4.5</version>

<release>2</release>

<vendor>www.dotnetblogengine.net</vendor>

<packager>parallels.com</packager>

</package>

<aps-catalog-url>http://apscatalog.com/</aps-catalog-url>

</download-package>

</aps>

</packet>

Dieses Paket lädt verschiedene Pakete mit Blog-Anwendungen aus dem Standard-APS-Katalog herunter. Es wird erwartet, dass jeweils die aktuellste Anwendung in den Katalog-Paketen heruntergeladen wird.

<packet version=”1.6.2.0”>

<aps>

<download-package>

<package>

<name>BlogEngine.NET</name>

</package>

</download-package>

<download-package>

<package>

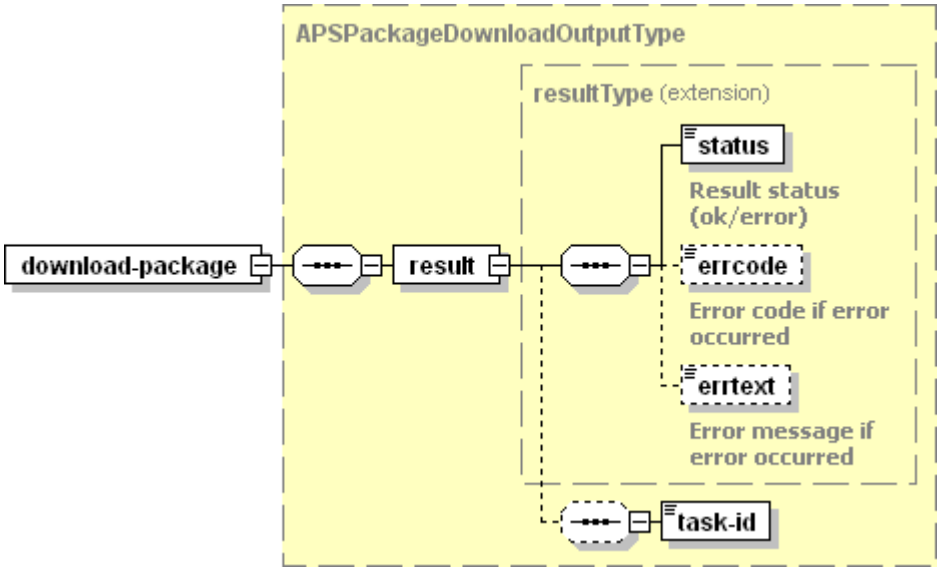
<name>WordPress</name>

</package>

```
    </download-package>
    <download-package>
    <package>
      <name>geeklog</name>
    </package>
    </download-package>
</aps>
</packet>
```

Struktur des Antwortpakets

Der **download-package**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **task-id**, *erforderlich,* falls die Operation erfolgreich abgeschlossen wird
  Gibt die ID des Download-Tasks an, die später verwendet werden kann, um den Download-Status abzufragen. Datentyp: *string*.

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<aps>
    <download-package>
    <result>
        <status>ok</status>
        <task-id>4</task-id>
    </result>
    </download-package>
</aps>
</packet>
```

Folgende Antwort wird empfangen, wenn die URL des APS-Katalog, die in der Abfrage angegeben wird, nicht in der Konfigurationsdatei des APS-Katalogs (auf Seite 146) aufgelistet wird. Es trifft auch auf die Fälle zu, in denen die URL in der Abfrage sich von der URL in der Konfigurationsdatei in kleinen Details unterscheidet. Zum Beispiel: Der URL fehlt der nachgestellte Schrägstrich, der aber in der Konfiguration des APS-Katalogs angegeben wurde.

```
<packet version=”1.6.2.0”>
<aps>
    <download-package>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>Catalog with URL “http://apscatalog.com” does not
  exist</errtext>
    </result>
    </download-package>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, wenn die URL des APS-Katalogs nicht aufgelöst werden kann:

```
<packet version=”1.6.2.0”>
<aps>
```

```
    <download-package>
    <result>
        <status>error</status>
        <errcode>1023</errcode>
  <errtext>Could not resolve host: apscatalog.co; No data record of
requested type</errtext>
    </result>
    </download-package>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, wenn die Konfigurationsdatei des APS-Katalogs (auf Seite 146) ein ungültiges Format aufweist:

```
<packet version=”1.6.2.0”>
<aps>
    <download-package>
    <result>
        <status>error</status>
        <errcode>1023</errcode>
     <errtext>APS Catalog configuration file is broken</errtext>
    </result>
    </download-package>
</aps>
</packet>
```

Status eines Download-Tasks abfragen

Um den Status des Download-Vorgangs eines Pakets abzufragen, erstellen Sie ein XML-Abfrage-Paket mit dem **aps/get-download-status**-Kontrollknoten:

<packet version="1.6.2.0">

<aps>

```
  <get-download-status>
...
  </get-download-status>
```

</aps>

</packet>

Die folgenden Statusmeldungen sind möglich:

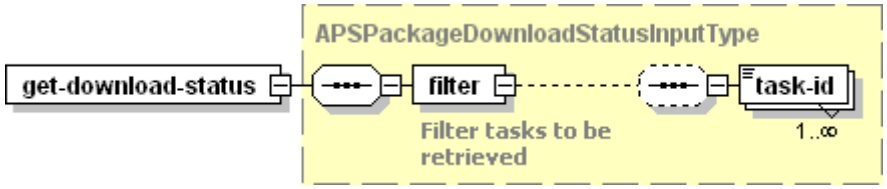
- `"finished/error"` - Der Download-Vorgang konnte nicht gestartet werden, da das Paket nicht in dem APS-Katalog gefunden werden konnte
- `"finished/package-id"` - Der Download-Vorgang wurde erfolgreich abgeschlossen
- `"in-progress/current"` - Das Paket wird momentan heruntergeladen
- `"in-progress"` ohne den untergeordneten ("child") Knoten `"current"` bedeutet, dass der Download-Vorgang des Pakets noch nicht gestartet wurde, sich aber in der Warteschlange befindet

In diesem Abschnitt:

Struktur des Abfrage-Pakets.....174
Beispiele für Abfragen.....176
Struktur des Antwortpakets.....177
Abfrage- und Antwort-Beispiele.....180

Struktur des Abfrage-Pakets

Der **get-download-package**-Knoten ist wie folgt strukturiert:



- **filter**, *erforderlich*

  Filtert Download-Vorgänge, deren Status abgefragt werden soll. Datentyp: none.

  Falls an dieser Stelle kein Wert definiert wurde, dann wird eine Liste mit allen Download-Tasks innerhalb der aktuellen Download-Transaktion wiedergegeben. Auf Windows wird für jeden Download-Task eine separate Transaktion erstellt. Die Download-Transaktion dauert so lange wie der Download-Task.

- **task-id**, *optional*

  Gibt die ID des Download-Tasks an (wiedergegeben als das Ergebnis der `<download-package>`-Operation). Datentyp: *id_type* (`common.xsd`).

Beispiele für Abfragen

Dieses Paket fragt den Status der aktuellen Download-Transaktion ab.

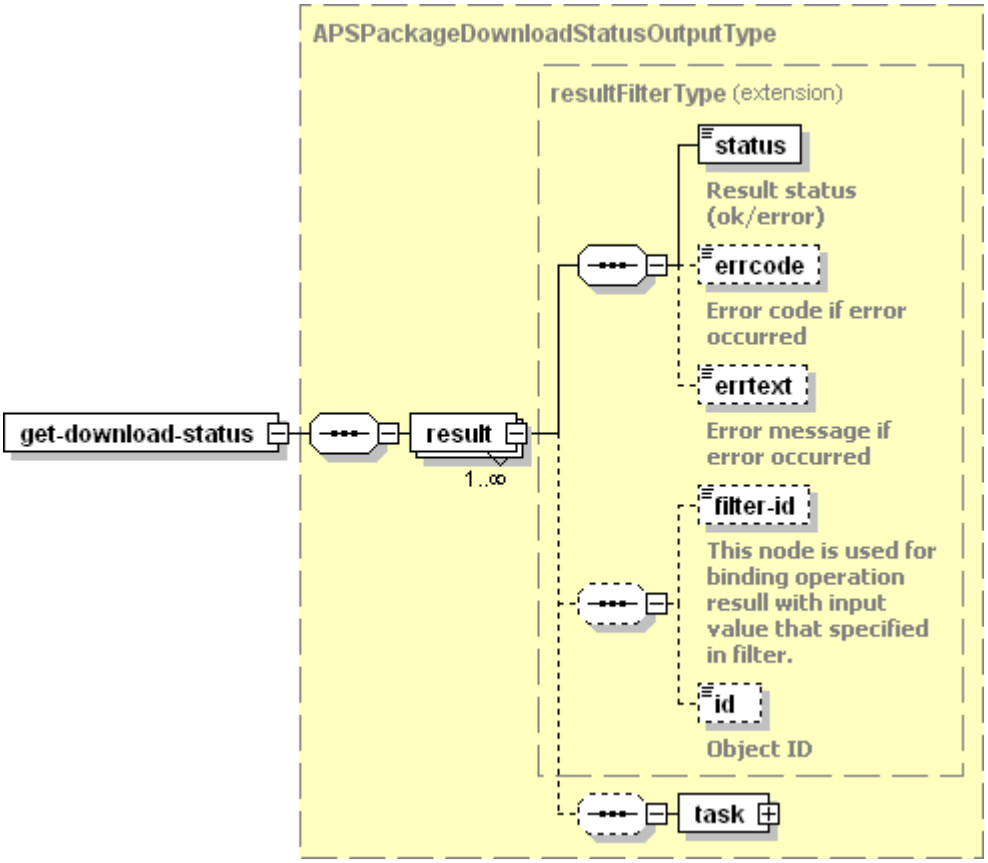
```
<packet version=”1.6.2.0”>
  <aps>
<get-download-status>
    <filter/>
</get-download-status>
  </aps>
</packet>
```

Dieses Paket fragt den Status der Download-Tasks mit den IDs 10,11 und 12 ab.

```
<packet version=”1.6.2.0”>
  <aps>
<get-download-status>
    <filter>
   <task-id>10</task-id>
   <task-id>11</task-id>
   <task-id>12</task-id>
    </filter>
</get-download-status>
  </aps>
</packet>
```

Struktur des Antwortpakets

Der **get-download-status**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*

  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*

  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt

  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Abfrage einen nicht leeren Filter enthält

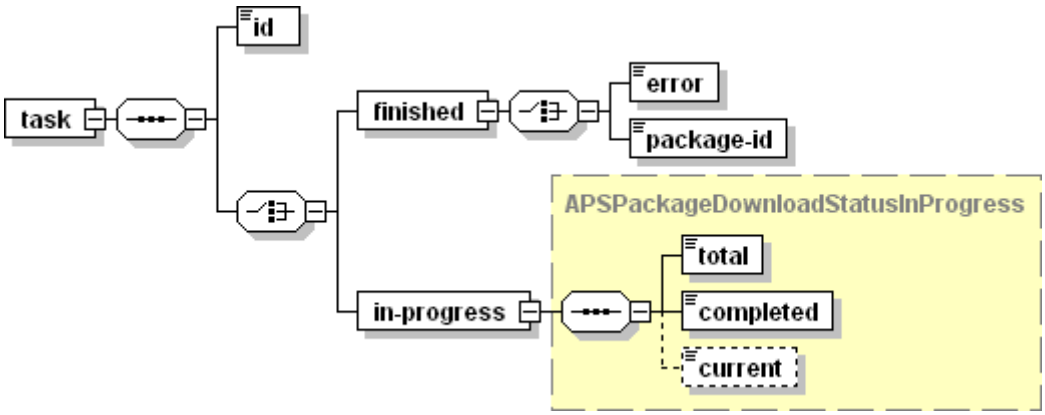
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **filter-id**, *erforderlich,* falls die Operation erfolgreich durchgeführt werden kann

  Gibt die ID wieder, nach der die Operation gefiltert wurde. Datentyp: *any.*
- **id**, *erforderlich,* falls die Operation erfolgreich abgeschlossen wird

  Gibt die ID des Download-Tasks an, dessen Status abgefragt wurde. Datentyp: *id_type* (`common.xsd`)
- **task**, *optional*

  Beinhaltet eine Reihe von Daten, die den Status des Download-Tasks beschreiben. Datentyp: none.

  Wie folgt strukturiert:



- **id**, erforderlich

  Gibt die ID des Download-Tasks an, dessen Status abgefragt wurde. Datentyp: *id_type* (`common.xsd`).

- **finished**, *optional*

  Gibt an, dass der Download-Task beendet wurde. Datentyp: none.

- **in-progress**, *optional*

  Gibt an, dass der Download Task in Bearbeitung ist oder sich in der Download-Warteschlange befindet, falls mehrere Pakete zum Download zur selben Zeit angegeben wurden. Datentyp: *APSPackageDownloadStatusInProgress* (`aps.xsd`).

- **error**, *optional*

  Gibt an, dass der Download mit einem Fehler beendet wurde. Beinhaltet eine Fehlerbeschreibung. Datentyp: none.

- **package-id**, *erforderlich*, wenn das Paket erfolgreich heruntergeladen wurde

  Gibt an, dass der Download erfolgreich abgeschlossen wurde und benennt die ID, die dem Paket auf Ihrem Server zugewiesen wurde. Datentyp: *id_type* (`common.xsd`).

- **total**, *erforderlich*

  Gibt die gesamte Anwendungsgröße in Bytes an. Datentyp: *integer*.

- **completed**, *erforderlich*

  Gibt die Größe des heruntergeladenen Teils an, auch in Bytes. Datentyp: *integer*.

- **current**, *optional*

  Gibt an, dass der Download-Task momentan durchgeführt wird im Gegensatz zu den Tasks, die in Download-Warteschlange warten. Datentyp: *string*.

Abfrage- und Antwort-Beispiele

Beispiel 1

Abfrage

Dieses Paket fragt den Status des Download-Tasks mit der ID 18 ab:

<packet version=”1.6.2.0”>

<aps>

<get-download-status>

<filter>
<task-id>18</task-id>
</filter>
</get-download-status>
</aps>

</packet>

Antwort A

Eine positive Antwort könnte wie folgt aussehen:

<packet version=”1.6.2.0”>
<aps>
<get-download-status>
<result>
<status>ok</status>
<filter-id>18</filter-id>
<id>18</id>
<task>
<id>18</id>
<finished>
<package-id>13</package-id>
</finished>
</task>
</result>

```
    </get-download-status>
</aps>
</packet>
```

Antwort B

Folgende Antwort wird empfangen, wenn ein Download-Task nicht gestartet wurde oder mit Fehlern beendet wurde.

```
<packet version="1.6.2.0">
<aps>
    <get-download-status>
    <result>
        <status>ok</status>
        <filter-id>18</filter-id>
        <id>18</id>
        <task>
        <id>18</id>
        <finished>
     <error>Cannot download package from catalog</error>
        </finished>
        </task>
    </result>
    </get-download-status>
</aps>
</packet>
```

Antwort C

Folgende negative Antwort wird empfangen, wenn ein Download-Task mit der angeforderten ID nicht existiert.

```
<packet version="1.6.2.0">
<aps>
    <get-download-status>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
     <errtext>Download task with id 18 not found</errtext>
     <filter-id>181</filter-id>
    </result>
```

```
    </get-download-status>
</aps>
</packet>
```

Beispiel 2

Abfrage

Dieses Paket fragt den Status aller Download-Tasks innerhalb einer aktuellen Download-Transaktion ab.

<packet version="1.6.2.0">

```
  <aps>
```

<get-download-status>

```
    <filter/>
```

</get-download-status>

```
  </aps>
</packet>
```

Beispiel A

Folgende Antwort wird von einem Linux-Server empfangen, wenn eine Transaktion mit drei Tasks ausgeführt wird: ein Task ist bereits beendet, einer ist in Bearbeitung und einer befindet sich in der Warteschlange.

```
<packet version="1.6.2.0">
<aps>
    <get-download-status>
    <result>
        <status>ok</status>
        <filter-id>10</filter-id>
        <id>10</id>
        <task>
        <id>10</id>
        <finished>
    <package-id>5</package-id>
```

```
        </finished>
        </task>
    </result>
    <result>
        <status>ok</status>
        <filter-id>11</filter-id>
        <id>11</id>
        <task>
        <id>11</id>
        <in-progress>
            <total>3584225</total>
            <completed>2458</completed>
            <current />
        </in-progress>
        </task>
    </result>
    <result>
        <status>ok</status>
        <filter-id>12</filter-id>
        <id>12</id>
        <task>
        <id>12</id>
        <in-progress>
            <total>45888236</total>
            <completed>0</completed>
        </in-progress>
        </task>
    </result>
    </get-download-status>
</aps>
</packet>
```

Antwort B

Folgende Antwort wird vom Windows-Server empfangen, wenn eine Transaktion ausgeführt wird:

```
<packet version=”1.6.2.0”>
<aps>
```

```
<get-download-status>
<result>
    <status>ok</status>
    <filter-id>38</filter-id>
    <id>38</id>
    <task>
    <id>38</id>
    <in-progress>
        <total>0</total>
        <completed>0</completed>
    </in-progress>
    </task>
</result>
</get-download-status>
</aps>
</packet>
```

Antwort C

Folgende Antwort wird empfangen, wenn keine Download-Transaktion durchgeführt wird.

```
<packet version=”1.6.2.0”>
<aps>
    <get-download-status>
    <result>
      <status>ok</status>
    </result>
    </get-download-status>
</aps>
</packet>
```

Installieren einer Anwendung

Bei der Installation einer Anwendung ist es erforderlich, dass Sie angeben, welches Paket als Quelle für die Installation verwendet werden soll. Das Paket wird entweder über die zugewiesene ID ermittelt, wenn das Paket in das Panel importiert wird oder über die Paketinformationen “package name-version-release-vendor-packager” (Paketname-Version-Release-Anbieter-Verpacker).

Wählen Sie einen der folgenden Schritte, um eine Paket-ID anzugeben:

- verwenden Sie die ID, die dem Paket beim Import in das Panel zugewiesen wurde (auf Seite 165)
- verwenden Sie die ID, die dem Paket beim erfolgreichen Download vom APS-Katalog zugewiesen (auf Seite 177)
- verwenden Sie die ID, die in den Informationen der verfügbaren Pakete angegeben wird (auf Seite 202)

Um die Paketinformationen “package name-version-release-vendor-packager” (Paketname-Version-Release-Anbieter-Verpacker) anzugeben, fragen Sie die Informationen zu den Paketen ab, die zur Installation zur Verfügung stehen (auf Seite 200).

In diesem Abschnitt:

Installieren einer Anwendung auf einer Domain oder Subdomain..... 185
Verfügbare Pakete abrufen 200

Installieren einer Anwendung auf einer Domain oder Subdomain

Um eine Anwendung zu installieren, erstellen Sie ein XML-Abfrage-Paket mit dem **aps/install**-Kontrollknoten:

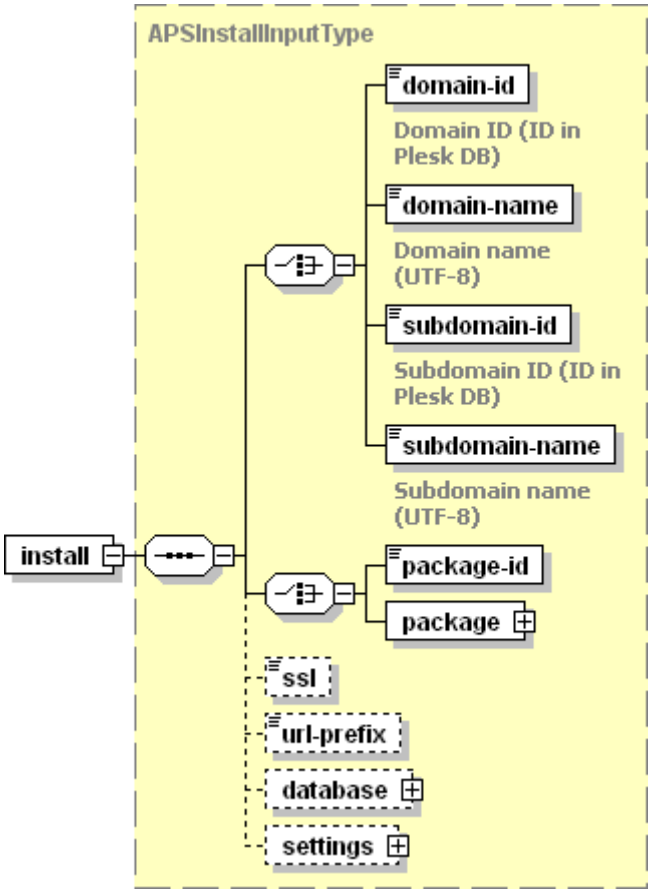
```
<packet version=”1.6.2.0”>
<aps>
  <install>
...
  </install>
</aps>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets..... 186
Beispiele für Abfragen..... 191
Struktur des Antwortpakets 193
Antwort-Beispiele 194

Struktur des Abfrage-Pakets

Der **install**-Knoten ist wie folgt strukturiert:



Entweder domain-id **oder** domain-name **oder** subdomain-id **oder** subdomain-name **ist *erforderlich*.**

- **domain-id**, *optional*

  Gibt die ID der Domain an, auf der die Anwendung installiert werden soll. Datentyp: *id_type* (`common.xsd`).

- **domain-name**, *optional*

  Gibt den Namen der Domain an, auf der die Anwendung installiert werden soll. Datentyp: *string*.

- **subdomain-id**, *optional*

  Gibt die ID der Subdomain an, auf der die Anwendung installiert werden soll. Datentyp: *id_type* (`common.xsd`).

- **subdomain-name**, *optional*

  Gibt den Namen der Subdomain an, auf der die Anwendung installiert werden soll. Datentyp: *string*.

Entweder **package-id** oder **package** ist *erforderlich*.

Hinweis: Während der **aps/get-packages-list**-Operation (auf Seite 200) werden die Informationen zu einem Paket ermittelt.

- **package-id**, *optional*

  Gibt die Paket-ID von dem Server an, von dem eine Anwendung installiert werden soll. Datentyp: *id_type* (`common.xsd`).

- **package**, *optional*

  Beinhaltet eine Reihe an Daten, die das Paket auf dem Server beschreiben, von dem aus eine Anwendung installiert werden soll. Datentyp: *APSIdentifierType* (`aps.xsd`), sehen Sie die Typdefinition unten.

- **ssl**, *optional*

  Gibt an, ob eine Anwendung über das HTTPS-Protokoll zur Verfügung steht oder nicht. Datentyp: *boolean*.

  Falls nicht angegeben, ist die Anwendung über HTTP verfügbar.

- **url-prefix**, *optional*

  Gibt die zur Domain/Subdomain zugehörige URL an, über welche die installierte Anwendung im Internet verfügbar sein wird. Datentyp: *APSUrlPrefixType* (`aps.xsd`).

  Falls nicht angegeben, wird das im Paket definierte Standardpräfix verwendet (`"/application/default-prefix"` -Element in dem Paket `APP-META.xml`-Datei).

- **database**, *optional*

  Beinhaltet eine Reihe an Daten, die die Eigenschaften einer Datenbank beschreiben, die für die Anwendung erstellt werden (sofern notwendig). Datentyp: `APSDatabaseInputType (aps.xsd)`, sehen Sie die Typdefinition unten.

  Falls für eine Anwendung eine Datenbank erforderlich ist und das **database**-Element nicht in der Abfrage angegeben wurde, wird diese automatisch mit dem Namen, Benutzer-Login und automatisch generierten Passwort erstellt.

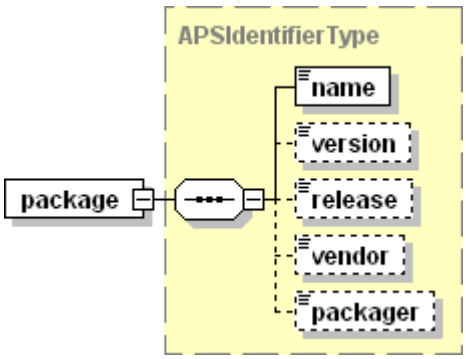
  Datenbankserver (MySQL, PostgreSQL oder Microsoft SQL Server) wird automatisch ausgewählt, um besser die Anwendungsanforderungen zu erfüllen.

- **settings**, *optional*

  Beinhaltet eine Auflistung mit Installationseinstellungen für die Anwendung. Datentyp: *APSSettingsInputType* (`aps.xsd`), sehen Sie die Typdefinition unten.

  Muss vorhanden sein, wenn die zu installierende Anwendung über Einstellungen verfügt (die Metadaten-Datei für das Anwendungspaket `APP-META.xml` beinhaltet mindestens ein Element "`//setting`").

Der **package** -Knoten wird durch den Typ *APSIdentifierType* (`aps.xsd`) dargestellt und seine grafische Darstellung ist wie folgt:



- **name**, *erforderlich*

  Gibt den Namen des Anwendungspakets an. Datentyp: *string*.

- **version**, *optional*

  Gibt die Version der Anwendung an. Datentyp: *string*.

- **release**, *optional*

  Gibt die Paket-Release an. Datentyp: *string*.

- **vendor**, *optional*

  Beinhaltet Informationen über den Anbieter der Anwendung. Datentyp: *string*.

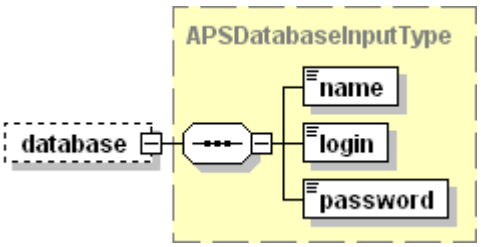
- **packager**, *optional*

  Gibt den Namen des Packagers (“Verpacker”) des Anwendungspakets an. Datentyp: *string*

Das Paket muss in einer gekürzten oder erweiterten Form angegeben werden:

- *reduced* beinhaltet nur den Namen der Anwendung und die aktuellste Version und Release, die installiert wurde
- *extended* beinhaltet alle Paketdetails: Name, Version, Release, Anbieter und Packager (“Verpacker”)

Der **database**-Knoten wird durch den Typ *APSDatabaseInputType* (`aps.xsd`) dargestellt und die grafische Darstellung ist wie folgt:



- **name**, *erforderlich*

  Gibt den Namen der Datenbank an. Sollte in dem System eindeutig sein. Datentyp: *string*.

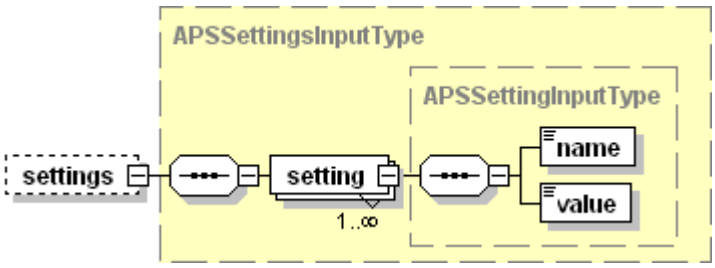
- **login**, *erforderlich*

  Gibt den Login-Namen des Datenbankbenutzers an. Sollte in dem System eindeutig sein. Datentyp: *string*.

- **password**, *erforderlich*

  Gibt das Passwort des Datenbankbenutzers an. Datentyp: *string*.

Der **settings**-Knoten wird durch den Typ *APSSettingsInputType* (`aps.xsd`) dargestellt und seine grafische Darstellung ist wie folgt:



- **setting**, *erforderlich*

  Beinhaltet eine Definition von Anwendungseinstellungen. Datentyp: *APSSettingInputType* (`aps.xsd`).

  Informationen zu den Namen von Anwendungseinstellungen und mögliche Werte sollten von der Metadaten-Datei des Anwendungspakets angefragt werden `APP-META.xml` (die `"//setting"`-Elemente).

  Die Abfrage muss das **setting**-Element für jede Anwendungseinstellung enthalten ohne den Standardwert in der Metadaten-Datei des Anwendungspakets.

- **name**, *erforderlich*

  Gibt die Einstellungs-ID an. Sollte der Einstellungs-ID in `APP-META.xml` (`"//setting[@id]"`) entsprechen. Datentyp: *string*.

- **value**, *erforderlich*

  Gibt den Einstellungs-Wert an. Datentyp: *string*.

Beispiele für Abfragen

Das folgende Paket installiert die Anwendung Serendipity auf der Domain example.com mit allen erforderlichen Parametern.

```
<packet version=”1.6.2.0”>
<aps>
<install>
  <domain-name>example.com</domain-name>
   <package>
   <name>serendipity</name>
   <version>1.1.2</version>
   <release>33</release>
   <vendor>s9y.org</vendor>
   <packager>parallels.com</packager>
   </package>

   <database>
   <name>s9DB</name>
   <login>s9user</login>
   <password>s9password</password>
   </database>

   <settings>
   <setting>
       <name>admin_p_name</name>
       <value>admin</value>
   </setting>
   <setting>
       <name>admin_name</name>
       <value>admin</value>
   </setting>
   <setting>
       <name>admin_password</name>
       <value>P4$$w0rd</value>
   </setting>
   <setting>
```

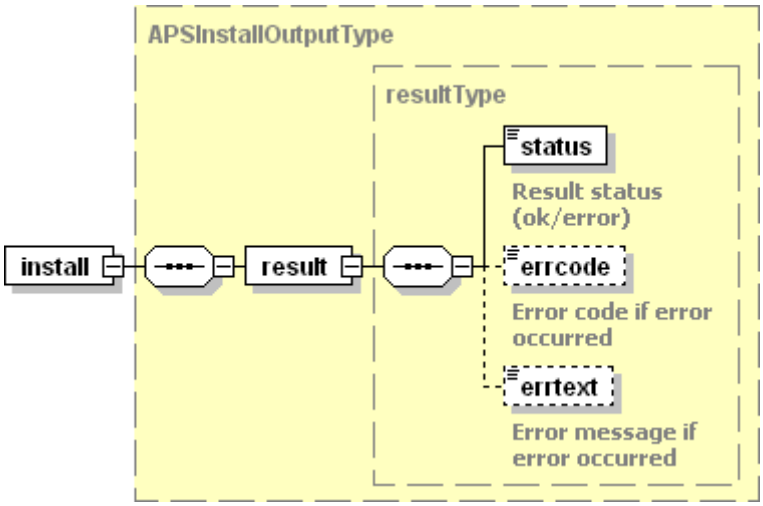
```
        <name>admin_email</name>
        <value>admin@example.com</value>
    </setting>
    <setting>
        <name>title</name>
        <value>Project X Blog</value>
    </setting>
    <setting>
        <name>blog_descr</name>
        <value>Project X Blog</value>
    </setting>
    <setting>
        <name>locale</name>
        <value>en-US</value>
    </setting>
    </settings>
</install>
</aps>
</packet>
```

Das folgende Paket installiert die Anwendung LinkPoint (Paket-ID auf dem Server: 5) auf einer Subdomain. Die folgende Abfrage ist nur möglich, wenn Anwendungen installiert werden, die keine Datenbank erfordern und für die es keine Einstellungen gibt.

```
<packet version="1.6.2.0">
<aps>
<install>
  <subdomain-name>subdomain.example.com</subdomain-name>
  <package-id>5</package-id>
</install>
</aps>
</packet>
```

Struktur des Antwortpakets

Der **install**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.

Operations-spezifische Fehler

1013 - Das angegebene Objekt (Domain, Subdomain, Paket) konnte nicht gefunden werden
1019 - Die Anwendungen kann mit den angegebenen Einstellungen nicht installiert werden
1100 - Die Anwendungsvoraussetzungen sind nicht ausreichend

Antwort-Beispiele

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<aps>
    <install>
    <result>
      <status>ok</status>
    </result>
    </install>
</aps>
</packet>
```

Fehlermeldungen: ‘1013 object not found’

1. Folgende negative Antwort wird empfangen, wenn eine angegebene Domain oder Subdomain nicht existiert.

```
<packet version=”1.6.2.0”>
<aps>
    <install>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
        <errtext>Domain not found.</errtext>
    </result>
    </install>
</aps>
</packet>
```

2. Folgende negative Antwort wird empfangen, wenn ein angefordertes Paket nicht auf dem Server gefunden werden konnte.

```
<packet version=”1.6.2.0”>
<aps>
    <install>
    <result>
        <status>error</status>
```

<errcode>1013</errcode>
<errtext>Package does not exist</errtext>
</result>
</install>
</aps>
</packet>

Fehlermeldungen: ‘1019 bad settings’

1. Folgende negative Antwort wird empfangen, wenn manche Anwendungseinstellungen nicht über Standardwerte verfügen und das Abfrage-Paket die Einstellungsspezifikation nicht enthält.

ABFRAGE-PAKET

<packet version=”1.6.2.0”>
<aps>
<install>
<domain-name>example.com</domain-name>
<package>
<name>AutoIndex</name>
</package>
</install>
</aps>
</packet>
ANTWORTPAKET
<packet version=”1.6.2.0”>
<aps>
<install>
<result>
<status>error</status>
<errcode>1019</errcode>
<errtext>Some errors occurred during installation:

Value of setting “admin_password” is invalid: “ is less than 1 characters long

Value of setting “user_password” is invalid: “ is less than 1 characters long</errtext>
</result>
</install>

```
</aps>
</packet>
```

2. Folgende Antwort wird empfangen, wenn die in der Abfrage definierten Anwendungseinstellungen nicht gegen die Datei `APP-META.xml` validiert werden konnten.

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
<aps>
    <install>
    <domain-id>10</domain-id>
    <package>
    <name>joomla</name>
    </package>
    <settings>
    <setting>
        <name>admin_name</name>
        <value>admin</value>
    </setting>
    <setting>
        <name>admin_password</name>
        <value>dfkjEEtbgg</value>
    </setting>
    <setting>
        <name>admin_email</name>
        <value>johndoe@example.com</value>
    </setting>
    <setting>
        <name>locale</name>
        <value>en-US</value>
    </setting>
    <setting>
        <name>title</name>
        <value>Joomla</value>
    </setting>
    </settings>
    </install>
```

</aps>

</packet>

ANTWORTPAKET

<packet version=”1.6.2.0”>

<aps>

<install>

<result>

<status>error</status>

<errcode>1019</errcode>

<errtext>Some errors occurred during installation:

The value of “locale” should be one of (en-GB, fr-FR, de-DE)</errtext>

</result>

</install>

</aps>

</packet>

3. Folgende negative Antwort wird empfangen, wenn das als URL-Präfix definierte Webverzeichnis bereits existiert und für einen anderen Content bereits verwendet wird:

<packet version=”1.6.2.0”>

<aps>

<install>

<result>

<status>error</status>

<errcode>1019</errcode>

<errtext>Some errors occurred during installation:

Destination directory already exists and is used by another Web application.</errtext>

</result>

</install>

</aps>

</packet>

4. Folgende negative Antwort wird empfangen, wenn die in dem Abfrage-Paket definierte Datenbank bereits existiert:

<packet version=”1.6.2.0”>

<aps>

<install>

<result>

<status>error</status>

```
      <errcode>1019</errcode>
    <errtext>Some errors occurred during installation:
```

Database with requested name already exists</errtext>

```
    </result>
    </install>
</aps>
</packet>
```

Fehlermeldung: ‘1100 requirements not met’

Folgende negative Antwort wird empfangen, wenn die Anwendungsvoraussetzungen nicht ausreichend erfüllt werden:

```
<packet version=”1.6.2.0”>
<aps>
    <install>
    <result>
        <status>error</status>
        <errcode>1100</errcode>
  <errtext>The following problems with requirements were found: Unable
to meet ASP.NET version requirements. ASP.NET version 2.0 is
erforderlich.; Unable to meet ASP.NET version requirements. ASP.NET
version 2.0 is erforderlich.; Unable to meet database server
requirements: Database server with type ‘mssql’ and version not less 9.0
not found</errtext>
    </result>
    </install>
</aps>
</packet>
```

Sonstige Fehlermeldungen

Folgende negative Antwort wird empfangen, wenn das Abfrage-Paket festlegt, dass die Anwendung über HTTPS (<ssl>true</ssl>) verfügbar sein soll, aber HTTPS auf der Domain/Subdomain nicht unterstützt wird.

ABFRAGE-PAKET

```
<packet version=”1.6.2.0”>
```

```
<aps>
    <install>
    <domain-id>2</domain-id>
    <package>
      <name>joomla</name>
    </package>
    <ssl>true</ssl>
    <settings>
    <setting>
        <name>admin_name</name>
        <value>admin</value>
    </setting>
    <setting>
        <name>admin_password</name>
        <value>dfkjEEtbgg</value>
    </setting>
    <setting>
        <name>admin_email</name>
        <value>johndoe@example.com</value>
    </setting>
    <setting>
        <name>locale</name>
        <value>en-GB</value>
    </setting>
    <setting>
        <name>title</name>
        <value>Joomla</value>
    </setting>
    </settings>
    </install>
</aps>
</packet>
```

ANTWORTPAKET

```
<packet version="1.6.2.0">
<aps>
    <install>
    <result>
```

```
        <status>error</status>
        <errcode>1023</errcode>
  <errtext>The IP address associated with this domain is already used by
another domain for SSL support implementation</errtext>
    </result>
    </install>
</aps>
</packet>
```

Verfügbare Pakete abrufen

Verwenden Sie die **aps/get-packages-list**-Operation, um Informationen zu Anwendungspaketen abzurufen, die zur Installation auf Domains/Subdomains zur Verfügung stehen:

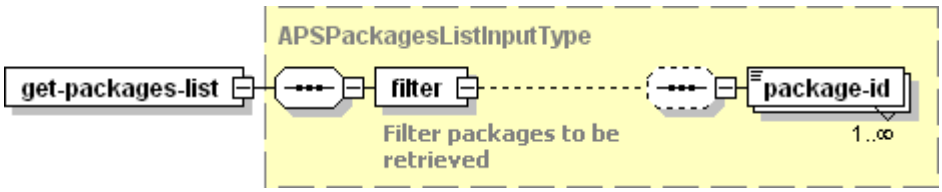
```
<packet version="1.6.2.0">
<aps>
  <get-packages-list>
...
  </get-packages-list>
</aps>
</packet>
```

In diesem Abschnitt:

Struktur des Abfrage-Pakets 200
Beispiele für Abfragen 201
Struktur des Antwortpakets 202
Abfrage- und Antwort-Beispiele 204

Struktur des Abfrage-Pakets

Der **get-packages-list**-Knoten ist wie folgt strukturiert:



- **filter**, *erforderlich*
  Filtert die Pakete, deren Informationen abgefragt werden sollen. Datentyp: none.
  Falls hier keine Angaben stehen (“empty”), werden Informationen zu allen auf dem Server verfügbaren APS-Paketen in dem Antwortpaket angegeben.
- **package-id**, *optional*
  Gibt die ID des Pakets an, dessen Informationen abgerufen werden sollen. Datentyp: *id_type* (`common.xsd`).

Beispiele für Abfragen

Dieses Paket ruft Informationen zu allen Anwendungspaketen ab.

<packet version=”1.6.2.0”>

<aps>

<get-packages-list>

<filter/>

</get-packages-list>

</aps>

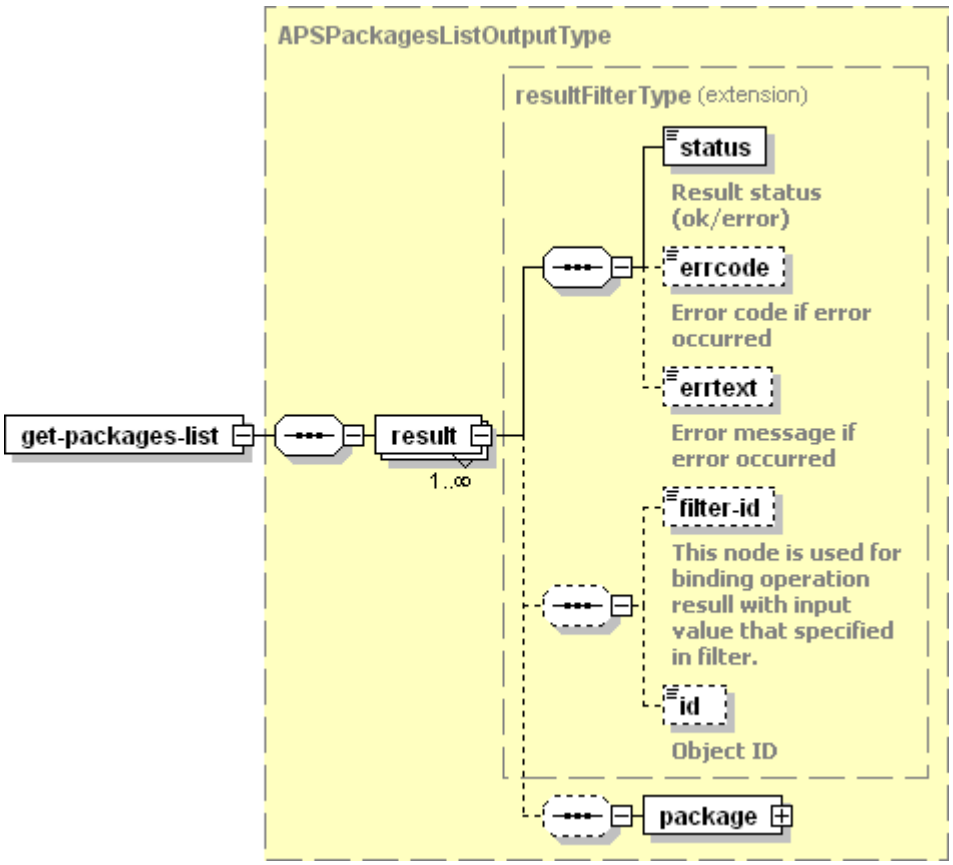
</packet>

Dieses Paket ruft Informationen zu allen Anwendungspaketen mit den IDs 11, 12 und 13 ab.

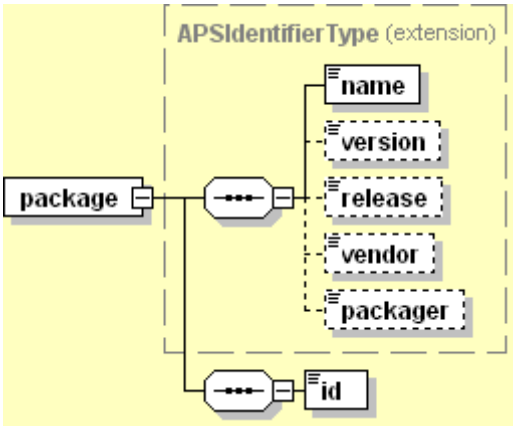
```
<packet version=”1.6.2.0”>
<aps>
<get-packages-list>
    <filter>
    <package-id>11</package-id>
    <package-id>12</package-id>
    <package-id>13</package-id>
    </filter>
</get-packages-list>
</aps>
</packet>
```

Struktur des Antwortpakets

Der **get-packages-list**-Knoten des ausgegebenen XML-Pakets ist wie folgt strukturiert:



- **result**, *erforderlich*
  Verpackt die vom Server abgerufene Antwort. Datentyp: *resultType* (`common.xsd`).
- **status**, *erforderlich*
  Gibt den Ausführungsstatus der Operation an. Datentyp: *string*. Zulässige Werte: ok | error.
- **errcode**, *erforderlich* , falls die Operation fehlschlägt
  Gibt den Fehlercode wieder. Datentyp: *integer*.
- **errtext**, *erforderlich* , falls die Operation fehlschlägt
  Gibt die Fehlermeldung wieder. Datentyp: *string*.
- **filter-id**, *erforderlich,* falls die Operation erfolgreich durchgeführt werden kann
  Gibt die Paket-ID an, nach der die Operation gefiltert wurde. Datentyp: *any.*
- **id**, *optional*
  Nicht angegeben.
- **package-id**, *erforderlich*, falls die Operation erfolgreich abgeschlossen wird
  Beinhaltet eine Reihe von Daten, die das Paket beschreiben. Datentyp: *APSIdentifierType* (`aps.xsd`).
  Wie folgt strukturiert:



- **name**, *erforderlich*
  Gibt den Namen des Anwendungspakets an. Datentyp: *string*.
- **version**, *optional*
  Gibt die Anwendungsversion an. Datentyp: *string*.
- **release**, *optional*
  Gibt die Anwendungs-Release an. Datentyp: *string*.
- **vendor**, *optional*
  Beinhaltet Informationen über den Anbieter der Anwendung. Datentyp: *string*.
- **packager**, *optional*
  Beinhaltet Informationen über den Packager (“Verpacker”) der Anwendung. Datentyp: *string*
- **id**, *erforderlich*
  Gibt die ID an, die dem Paket von dem Server zugewiesen wurde. Datentyp: *id_type* (`common.xsd`).

Abfrage- und Antwort-Beispiele

Beispiel 1

Abfrage

Dieses Paket ruft Informationen zu allen APS-Paketen ab.

<packet version=”1.6.2.0”>

```
<aps>
```

<get-packages-list>

```
<filter/>
```

</get-packages-list>

```
</aps>
```

</packet>

Beispiel A

Eine positive Antwort vom Server sieht wie folgt aus:

```
<packet version=”1.6.2.0”>
<aps>
    <get-packages-list>
    <result>
        <status>ok</status>
        <filter-id>1</filter-id>
        <package>
        <name>phpWiki</name>
        <version>1.3.11</version>
        <release>44</release>
        <vendor>phpwiki.sourceforge.net</vendor>
        <packager>parallels.com</packager>
        <id>1</id>
        </package>
    </result>
```

```
<result>
    <status>ok</status>
    <filter-id>2</filter-id>
    <package>
    <name>PinnacleCart</name>
    <version>3.6.1</version>
    <release>570</release>
    <vendor>www.pinnaclecart.com</vendor>
    <packager>www.pinnaclecart.com</packager>
    <id>2</id>
    </package>
</result>
</get-packages-list>
</aps>
</packet>
```

Antwort B

Folgende Antwort wird empfangen, wenn keine APS-Pakete auf dem Server vorhanden sind:

```
<packet version="1.6.2.0">
<aps>
    <get-packages-list>
    <result>
      <status>ok</status>
    </result>
    </get-packages-list>
</aps>
</packet>
```

Beispiel 2

Dieses Paket ruft Informationen zu allen APS-Paketen mit den IDs 12 und 13 ab.

```
<packet version="1.6.2.0">
<aps>
<get-packages-list>
    <filter>
    <package-id>12</package-id>
```

```
    <package-id>13</package-id>
    </filter>
</get-packages-list>
</aps>
</packet>
```

Folgende negative Antwort wird empfangen, falls das Paket mit der angeforderten ID nicht existiert:

```
<packet version=”1.6.2.0”>
<aps>
    <get-packages-list>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
        <errtext>Package does not exist</errtext>
        <filter-id>12</filter-id>
    </result>
    <result>
        <status>error</status>
        <errcode>1013</errcode>
        <errtext>Package does not exist</errtext>
        <filter-id>13</filter-id>
    </result>
    </get-packages-list>
</aps>
</packet>
```

KAPITEL 6

Einrichten über CLI

Dieses Kapitel konzentriert sich auf das Einrichten des Panels über die Befehlszeilenschnittstelle des Panels (Abk. CLI, Command Line Interface).

Der Abschnitt **Über die Panel-Befehlszeilenschnittstelle (CLI)** (auf Seite 208) gibt einen Einblick in die CLI-Basics.

Der Abschnitt **Einrichten des Panels** (auf Seite 209) beschreibt im Detail, welche Befehle ausgeführt werden sollten, um die Panel-Setup-Operationen durchzuführen.

In diesem Kapitel:

Über die Panel-Befehlszeilenschnittstelle (CLI) 208
Einrichten des Panels 209

Über die Panel-Befehlszeilenschnittstelle (CLI)

Die Befehlszeilenschnittstelle (Abk. CLI, Command Line Interface) von Parallels Small Business Panel bietet eine Befehlszeilenoberfläche über die ein Panel-Administrator die Mehrzahl der Verwaltungsaufgaben durchführen kann.

Die Panel-Befehlszeilendienstprogramme können vom root-Benutzer von der Konsole oder vom Skript verwendet werden. Um die Dienstprogramme auszuführen, benötigen Sie “root”/”psaadm”- (auf Linux/Unix) oder “Administrator” (auf Windows)-Rechte.

Parallels Small Business Panel für Linux/Unix-CLI unterstützt lokalisierte Namen von Business-Objekten in derselben Art und Weise wie das Panel an sich. Deshalb können Sie Kunden mit Kontaktnamen mithilfe eines Gebietsschemas Ihrer Wahl erstellen. Verwenden Sie die LANG-Umgebungsvariable, um ein bestimmtes Gebietsschema zu konfigurieren.

Nach dem erfolgreichen Ausführen geben Dienstprogramme den Code `0` an. Falls ein Fehler auftritt, geben Dienstprogramme den Code `1` an und zeigen eine Beschreibung des aufgetretenen Problems auf `stderr` an.

Hinweis: Alle Eingabeparameter für die Dienstparameter müssen in der UTF-8-Kodierung angegeben werden. Dies ist unbedingte Voraussetzung, damit das Panel stabil läuft.

Speicherorte von Dienstprogrammen

Auf Linux/Unix:

`/opt/psa/bin` auf Debian-Systemen

`/usr/local/psa/bin` auf nicht-Debian Linux und Unix

Auf Windows:

`%plesk_cli%`

Einrichten des Panels

Der Abschnitt beschreibt im Detail, welche Befehle ausgeführt werden sollten, um die Panel-Setup-Operationen durchzuführen.

In diesem Abschnitt:

Initialisierung des Panels 210
Installation des Lizenzschlüssels 211
Branding des Panels 211
DNS-Konfiguration 213
Installation von SSL-Zertifikaten 220
Erstellen von Domains 222
Erstellen von Subdomains 223
Verfügbare APS-Kataloge definieren 224
Installieren von Anwendungen 225

Initialisierung des Panels

Um das Panel über die Befehlszeilenschnittstelle (CLI) zu initialisieren, verwenden Sie bitte das Dienstprogramm `init_conf`.

Befehl

Der folgende Befehl initialisiert das Panel:

init_conf—init -email <email> -passwd <password> [-locale <locale-code>]

Der `<locale-code>` folgt dem RFC 1766-Standard in dem Format "<*languagecode2*>-<*country/regioncode2*>", wobei <*languagecode2*> ein zweistelliger Code bestehend aus Kleinbuchstaben ist, der von ISO 639-1 abgeleitet wurde, und <*country/regioncode2*> ist ein zweistelliger Code bestehend aus Großbuchstaben, der von ISO 3166 abgeleitet wurde. Beispiel: Der Ländercode für Deutsch ist (Germany) "de-DE".

Beispiele

1. Mit dem folgenden Befehl initialisieren Sie das Panel und konfigurieren das Administrator-Passwort "jskekekHTD" bzw. die E-Mail-Adresse "johndoe@example.com".

Auf Linux/Unix:

```
./init_conf—init -passwd jskekekHTD -email johndoe@example.com
```

Auf Windows:

```
init_conf—init -passwd jskekekHTD -email johndoe@example.com
```

2. Mit dem folgenden Befehl initialisieren Sie das Panel, konfigurieren das Administrator-Passwort "P4$$w0rd" und die E-Mail-Adresse "fischer@example.com" und für das Panel den Ländercode "Deutsch (Deutschland)".

Auf Linux/Unix:

```
./init_conf—init -passwd 'P4$$w0rd' -email fischer@example.com -locale de-DE
```

Auf Windows:

```
init_conf—init -passwd 'P4$$w0rd' -email fischer@example.com -locale de-DE
```

Installation des Lizenzschlüssels

Um einen Panel-Lizenzschlüssel über die Panel-Befehlszeilenschnittstelle (CLI) zu installieren, sollten Sie diesen zuerst in den Computer mit dem Panel hochladen und dann mit dem `license`-Dienstprogramm installieren.

Befehl

Um eine Lizenz zu installieren, führen Sie einen Befehl in dem folgenden Format aus:

license <--install|-i> <path-to-key-file>

Beispiele

1. Auf Linux/Unix:

```
./license -i /tmp/psmbfu10key.xml
```

2. Auf Windows:

```
license.exe -i "D:\keys\psmbfw10key.xml"
```

Branding des Panels

Zur Konfiguration des Panel-Brandings verwenden Sie das `branding`-Dienstprogramm.

In diesem Abschnitt:

Ändern des Logos 212
Ändern der Support-URL 213

Ändern des Logos

Um das Panel-Logo und die URL zu ändern, sollten Sie die folgenden zwei Schritte durchführen:

1. Führen Sie einen Befehl in dem folgendem Format aus, um das Logo ändern zu können und eine Bilddatei hochzuladen:

   ```
   branding <--use-custom-logo|-u> true -path <path_to_file>
   ```

2. Mithilfe eines Befehls in dem folgenden Format können Sie eine benutzerdefinierte URL einrichten:

   ```
   branding <--change-logo-url|-l> -url <URL>
   ```

Beispiele

1.Der folgende Befehl ermöglicht die Anpassung des Logos und verwendet als Logo die Bilddatei `my-own-cool-logo.gif`, die sich in dem Verzeichnis `temp` befindet.

Auf Linux/Unix:

```
./branding -u true -path temp/my-own-cool-logo.gif
```

Auf Windows:

```
branding.exe -u true -path D:\temp\my-own-cool-logo.gif
```

2. Mithilfe des folgenden Befehls wird festgelegt, dass sich die URL http://example.com/about beim Anklicken des Panel-Logos öffnet.

Auf Linux/Unix:

```
./branding -l -url http://example.com/about
```

Auf Windows:

```
branding.exe -l -url http://example.com/about
```

Ändern der Support-URL

Um den Support-Link zu ändern (eine URL, die sich beim Anklicken des Support-Buttons öffnet), verwenden Sie einen Befehl in dem folgenden Format:

branding <--change-support-url|-c> -url <*URL*>

Beispiele

Über den folgenden Befehl öffnet sich ein benutzerdefiniertes Support-Anfrageformular, wenn ein Benutzer auf den Support-Button klickt https://support.example.com/submit-ticket.

Auf Linux/Unix:

```
./branding -c -url https://support.example.com/submit-ticket
```

Auf Windows:

```
branding.exe -c -url https://support.example.com/submit-ticket
```

DNS-Konfiguration

Um den DNS-Dienst des Panels über die CLI zu konfigurieren (auch um es abzuschalten), verwenden Sie das `server_dns`-Dienstprogramm.

In diesem Abschnitt:

DNS-Dienst deaktivieren 213
Ändern des SOA-Record-Templates 214
Einrichten von Templates für Ressourceneinträge 215

DNS-Dienst deaktivieren

Um den DNS-Dienst des Panels zu deaktivieren, führen Sie den folgenden Befehl aus:

server_dns <--update-server|-u> -status disabled

Auf Linux/Unix:

```
./server_dns -u -status disabled
```

Auf Windows:

```
server_dns.exe -u -status disabled
```

Ändern des SOA-Record-Templates

Um das SOA-Record-Template zu ändern, geben Sie den Befehl in dem folgenden Format ein:

server_dns—update-soa [options]

Verfügbare Optionen

Option	Parameter	Beschreibung
-soa-ttl	<number>[S|M|H|D|W]	Definiert den TTL-Intervall (Time-to-Live). Das Panel legt als Standardwert einen Tag fest.
-soa-refresh	<number>[S|M|H|D|W]	Definiert den Refresh-Intervall (Aktualisieren). Das Panel legt als Standardwert drei Stunden fest.
-soa-retry	<number>[S|M|H|D|W]	Definiert den Retry-Intervall (Wiederholen). Das Panel legt als Standardwert eine Stunde fest.
-soa-expire	<number>[S|M|H|D|W]	Definiert den Expire-Intervall (Ablaufen). Das Panel legt als Standardwert eine Woche fest.
-soa-minimum	<number>[S|M|H|D|W]	Definiert einen Mindestwert (Minimum). Das Panel legt als Standardwert drei Stunden fest.
-soa-serial-format	timestamp|yyyymmddnn	Definiert das Format einer SOA-Record-Seriennummer: UNIX-Timestamp oder empfohlen von IETF und RIPE. Wird nur auf Linux/Unix unterstützt.

Beispiele

1. Der folgende Befehl setzt den TTL-Wert auf *5* Stunden, den Refresh-Wert (Aktualisieren) auf *4* Stunden, den Retry-Wert (Wiederholen) auf *10* Sekunden, den Expire-Wert (Ablaufen) auf *2* Wochen und den Mindestwert auf *2* Tage.

Auf Linux/Unix:

```
./server_dns—update-soa -soa-ttl 5H -soa-refresh 4H -soa-retry 10S -soa-expire 2W -soa-minimum 2D
```

Auf Windows:

```
server_dns.exe—update-soa -soa-ttl 5H -soa-refresh 4H -soa-retry 10S -soa-expire 2W -soa-minimum 2D
```

2. Der folgende Befehl ändert das Format der SOA-Record-Seriennummer auf das Format, das von IETF und RIPE empfohlen wird:

```
./server_dns—update-soa -soa-serial-format yyyymmddnn
```

Einrichten von Templates für Ressourceneinträge

Sie können eine der folgenden zwei Möglichkeiten wählen, um über die Befehlszeilenschnittstelle (CLI) des Panels das DNS-Zonen-Template zu ändern:

1. **Hinzufügen eines Eintrags (Record)** Leichte Modifikation, indem Sie einen einzelnen Eintrag mit einem Befehl hinzufügen
2. **Überschreiben eines Zone-Templates**. Vollständige Modifikation, indem Sie alle Ressourceneinträge des Zonen-Templates entfernen und neue Einträge mit einem Befehl verfassen.

In diesem Abschnitt:

Hinzufügen eines Eintrags (Record) 215
Records im DNS-Zonen-Template überschreiben 218

Hinzufügen eines Eintrags (Record)

Um einen einzelnen Ressourceneintrag zu dem DNS-Zonen-Template hinzuzufügen, verwenden Sie den folgenden Befehl:

server_dns <--add|-a> <*RR-Parameter*>

Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-a	<subdomain_name>|""	Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden.	So fügen Sie einen neuen A-Ressourceneintrag *sample.<domain>* (definiert für den Eintrag einen Platzhalter für eine Domain-IP-Adresse) zu dem Domain-DNS-Zone-Template hinzu: **server_dns -a -a sample -ip "<ip>"**
-ip	<IP_address>|"<ip>"	Definiert eine IP-Adresse oder einen Platzhalter für eine Domain-IP-Adresse zu Erstellung eines A-Templates für den Ressourceneintrag.	

NS-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-ns	<subdomain_name>|""	Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden.	So fügen Sie einen neuen NS-Ressourceneintrag hinzu, der "ns.<domain>." als Nameserver

Parameter	Argument	Beschreibung	Beispiel
-nameserver	<domain_name>|"[<subdomain_name>.]<domain>"	Gibt den Namen eines Nameservers oder eines Platzhalters zur Erstellung eines NS-Ressourceneintrags-Template.	für den Host *<domain>* definiert: **server_dns -a -ns "" -nameserver "ns.<domain>"**

MX-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-mx	<subdomain_name>|""	Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden.	So fügen Sie einen neuen MX-Ressourceneintrag hinzu, der *mail.<domain>* als Mailserver für *<domain>* mit der Priorität *15* festlegt: **server_dns -a -mx "" -mailexchanger "mail.<domain>" -priority 15**
-mailexchanger	<domain_name>|"[<subdomain_name>.]<domain>"	Gibt den Namen eines Mail-Exchangers oder eines Platzhalters an.	
-priority	<number>	Gibt die Priorität des Mail-Exchangers an.	

CNAME-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-cname	<subdomain_name>|""	Gibt den Namen des Host an, für den die Eintragseinstellungen übernommen werden.	So fügen Sie einen neuen CNAME-Ressourceneintrag hinzu, der ns.<domain> als Alias für www.<domain> festlegt: **server_dns -a -cname www -canonical "ns.<domain>"**
-canonical	<domain_name>|[<subdomain_name>.]<domain>"	Gibt einen kanonischen Namen oder einen Platzhalter zur Erstellung eines CNAME-Record-Templates an.	

PTR-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-ptr	<subdomain_name>|""	Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden.	So fügen Sie einen neuen PTR-Eintrag hinzu, der <domain> zu dem Domain Name Pointer für

Parameter	Argument	Beschreibung	Beispiel
-subnet	<IP_address>/<subnet_mask>|"<ip>"/<subnet_mask>	Gibt eine IP-Adresse und eine Subnetzmaske oder einen Platzhalter zur Erstellung eines PTR-Record-Templates an.	das Subnetz *"<ip>"/16* macht: **server_dns -a -ptr "" -subnet "<ip>"/16**

SRV-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-srv	""|<subdomain_name>	Gibt den Namen des Hosts an, für den die Record-Einstellungen übernommen werden.	So fügen Sie einen SRV-Record für *<domain>* zum DNS-Zonen-Template hinzu, der SIP-Protokoll-Verbindungen verarbeitet, die auf den Server *sipserver.sample.com* verweisen am *TCP*-Port *5060*. Die Priorität hier ist *0* und das Gewicht ist *5*: **server_dns—add -srv "" -srv-service sip -srv-target-host sipserver.sample.com. -srv-protocol TCP -srv-port 5060 -srv-priority 0 -srv-weight 5**
-srv-priority	[0-50]	Gibt die Priorität des SRV-Records (0 - höchste Priorität, 50 - die niedrigste).	
-srv-weight	[0-50]	Gibt das relative Gewicht des SRV-Records unter den Records mit derselben Priorität an (0 - niedrigste, 50 - höchste).	
-srv-port	[0-65535]	Gibt den Port an, auf dem der Service zur Verfügung gestellt wird.	
-srv-target-host	<host>	Gibt den kanonischen Hostnamen der Maschine an, die den Service bereitstellt.	
-srv-protocol	TCP|UDP	Gibt den Serviceprotokoll an.	
-srv-service	<service>	Gibt den symbolischen Namen des Services an.	

TXT-Record-Parameter und Beispiel

Parameter	Argument	Beschreibung	Beispiel
-txt	<text>|""	Gibt eine Textbeschreibung an	Um den DNS.Record im TXT-Typ

Parameter	Argument	Beschreibung	Beispiel
-domain	<subdomain_name>|""	Gibt den Namen des Hosts an, für den die Eintragseinstellungen übernommen werden.	anzugeben, geben Sie folgendes *"This record is used to implement the Senders Policy Framework and DomainKeys specifications"* für den Host policy.<domain>an: **server_dns -a -txt "This record is used to implement the Senders Policy Framework and DomainKeys specifications" -domain policy**

Records im DNS-Zonen-Template überschreiben

Um ein DNS-Zonen-Template zu überschreiben, verwenden Sie den Befehl im folgenden Format:

server_dns <--set|-s> "<record_1>;<record_2>;...<record_N>"

wobei das *<record>*-Format ist: "*<type>*,*<host>*,*<value>*[,*<option>*]".

Das heißt das *<record>*-Format ist wie folgt für jeden Ressourceneintragstyp, der in dem Panel unterstützt wird:

NS,<<subdomain-prefix>|>,<<name-server-FQDN>|<prefix>.<domain>.>

A,<<subdomain-prefix>|>,<<IP-address>|<ip>>

MX,<<subdomain-prefix>|>,<<mail-exchanger-name>|<prefix>.<domain>.>,<priority>

PTR,<<IP-address>|<ip>>,<<subdomain-prefix>|>,<subnet_mask>

CNAME,<<subdomain-prefix>|>,<<canonical_domain_name>|<domain>.>

SRV,<_service>.<_tcp|_udp>.<<subdomain-prefix>|>,<target_host>,<priority> <weight <port>

TXT,<<*subdomain-prefix*>|>,<*text*>

Hinweis: Bitte beachten Sie, dass die Platzhalter für Befehle, die durch echte Werte in echten Befehlen ersetzt werden, in kursiv angegeben sind (z.B., "*<subdomain-prefix>*"), während die DNS-Template-Platzhalter, die in dem Befehl platziert werden soll, werden in Roman-Schreibweise angegeben sind (z.B., "<domain>.").

Beispiel

Der Beispiel-Befehl fügt die folgende DNS-Zonen-Template-Datei zu dem Panel hinzu (aus Darstellungsgründen haben wir den ".<domain>."-Teil zu Hosts wie z.B. "ns.<domain>." hinzugefügt, welche in echten Zonen-Dateien normalerweise nur "ns" wären):

```
HOST                RR type      VALUE

<Domain>.           NS           ns.<domain>.
ns.<Domain>.        A <IP>
<Domain>.           A <IP>
webmail.<Domain>.   A <IP>
<Domain>.           MX           10 mail.<domain>.
mail.<Domain>.      A <IP>
ftp.<domain>.       CNAME        <domain>.
<ip>/24             PTR          <domain>.
<Domain>.           TXT          f1 +a +mx -all
_SIP._tcp.megadomain.<domain>.       SRV    5 25 12 example.com.
```

Auf Linux/Unix:

```
./server_dns -s
'NS,,ns.<domain>.;A,ns,<ip>;A,,<ip>;A,webmail,<ip>;MX,,mail.<domain>.,10;A,
mail,<ip>;CNAME,ftp,<domain>.;PTR,<ip>,,24;TXT,,v=spf1 +a +mx
-all;SRV,_SIP._tcp.megadomain,example.com,5 25 12;'
```

Auf Windows:

```
server_dns.exe -s
"NS,,ns.<domain>.;A,ns,<ip>;A,,<ip>;A,webmail,<ip>;MX,,mail.<domain>.,10;A,
mail,<ip>;CNAME,ftp,<domain>.;PTR,<ip>,,24;TXT,,v=spf1 +a +mx
-all;SRV,_SIP._tcp.megadomain,example.com,5 25 12;"
```

Installation von SSL-Zertifikaten

Um ein SSL-Zertifikat über die Befehlszeilenschnittstelle (CLI) zu installieren, verwenden Sie das `certificate`-Dienstprogramm. Die Installation wird in zwei Phasen durchgeführt:

1. Import des Zertifikats in das Panel.
2. Zuweisen des Zertifikats zu einer IP-Adresse.

   Bitte beachten Sie, dass die Zertifikatsbestandteile zuvor auf die Maschine mit dem Panel hochgeladen werden müssen.

In diesem Abschnitt:

Import des Zertifikats 221
Zuweisen des Zertifikats zu einer IP-Adresse. 222

Import des Zertifikats

Um ein Zertifikat zu importieren, führen Sie den Befehl in dem folgenden Format aus:

certificate <--create|-c> <name> -admin -key-file <private-key-file> -cert-file <certificate-file> [-cacert-file <CA-certificate-file>] [-csr-file <certificate-signing-request-file>]

Beispiele

1. Der folgende Befehl importiert das SSL-Zertifikat mit der privaten Schlüssel-Datei `/usr/local/keys/keyfile.key` und der Zertifikatsdatei `/usr/local/cert/certfile.cert`. Nach dem Import heißt das Zertifikat “Site Certificate”.

Auf Linux/Unix:

```
./certificate—create “Site Certificate” -admin -key-file
/usr/local/keyfile.key -cert-file /usr/local/cert/certfile.cert
```

oder

```
./certificate -c “Site Certificate” -admin -key-file /usr/local/keyfile.key
-cert-file /usr/local/cert/certfile.cert
```

Auf Windows (die Dateien für diese Zertifikatsbestandteile befinden sich im Verzeichnis `E:\temp\`):

```
certificate—create “Site Certificate” -admin -key-file E:\temp\keyfile.key
-cert-file E:\temp\certfile.cert
```

oder

```
certificate -c “Site Certificate” -admin -key-file E:\temp\keyfile.key
-cert-file E:\temp\certfile.cert
```

2. Der folgende Befehl importiert das SSL-Zertifikat mit der privaten Schlüssel-Datei `/usr/local/keys/keyfile.key;` die Zertifikatsdatei ist `/usr/local/cert/certfile.cert,` die CA-Zertifikatsdatei ist `/usr/local/cert/cacert.cert` und die entsprechende CSR-Datei (Certificate Signing Request) ist /`usr/local/requests/csreq.csr`. Nach dem Import heißt das Zertifikat “Site Certificate”.

Auf Linux/Unix:

```
./certificate -c “Site Certificate” -domain example.com -key-file
/usr/local/keys/keyfile.key -cert-file /usr/local/cert/certfile.cert
-cacert-file /usr/local/cert/cacert.cert -csr-file
/usr/local/requests/csreq.csr
```

Auf Windows (die Dateien für diese Zertifikatsbestandteile befinden sich im Verzeichnis `E:\ssl\`):

```
certificate -c “Site Certificate” -domain example.com -key-file
E:\ssl\keys\keyfile.key -cert-file E:\ssl\cert\certfile.cert -cacert-file
E:\ssl\cert\cacert.cert -csr-file E:\ssl\requests\csreq.csr
```

Zuweisen des Zertifikats zu einer IP-Adresse.

Um ein importiertes Zertifikat zu einer IP-Adresse zu zuweisen, verwenden Sie den Befehl im folgenden Format:

certificate <--assign-cert|-ac> <*name*> -admin -ip <*ip-address*>

Beispiel

Der folgende Befehl weist das SSL-Zertifikat mit dem Namen “Site Certificate” der IP-Adresse 192.0.2.78 zu.

Auf Linux/Unix:

```
./certificate -ac “Site Certificate” -ip 192.0.2.78
```

Auf Windows:

```
certificate -ac “Site Certificate” -ip 192.0.2.78
```

Erstellen von Domains

Um einen Domain-Account zu erstellen, verwenden Sie das `domain`-Dienstprogramm.

Befehl

Mit dem folgenden Befehl erstellen Sie einen Domain-Account:

domain <--create|-c> <domain-name> -hosting true -ip <ip-address> -login <FTP-login> -passwd <FTP-password>

wobei *FTP-login>* und <*FTP-password>* den Login und das Passwort des FTP-Accounts enthalten, der zum Hochladen und zum Bearbeiten des Webcontents der auf der erstellten Domain gehosteten Website verwendet wird.

Beispiel

Der folgende Befehl erstellt die Domain *example.com*, die auf der IP-Adresse *192.0.2.33* gehostet wird, mit den FTP-Account-Anmeldeinformationen “johndoe” und “P4$$w0rd”.

Auf Linux/Unix:

```
./domain—create example.com -hosting true -ip 192.0.2.33 -login johndoe -passwd P4$$w0rd
```

Auf Windows:

```
domain.exe—create example.com -hosting true -ip 192.0.2.33 -login johndoe -passwd P4$$w0rd
```

Erstellen von Subdomains

Um einen Subdomain-Account zu erstellen, verwenden Sie das `subdomain`-Dienstprogramm.

Erstellen von regulären Subdomains

Befehl

Mit dem folgenden Befehl erstellen Sie eine Subdomain:

subdomain <--create|-c> <subdomain-prefix> -domain <parent-domain-name> [-ftp_user native -login <FTP-login> -passwd <FTP-password>]

wobei der Teil

- `ftp_user native -login` *<FTP-login>* `-passwd` *<FTP-password>*

verantwortlich ist, für das Erstellen eines neuen Subdomain-spezifischen FTP-Accounts (im Gegensatz zu dem Standardverhalten, wenn der FTP-Account der übergeordneten Domain ("Parent Domain") auch zur Verwaltung der Subdomains verwendet wird).

Beispiele

1. Der folgende Befehl erstellt die Subdomain `mail.example.com` und der FTP-Account, der zur Verwaltung des Inhalts verwendet wird, ist derselbe wie für die übergeordnete ("Parent") Domain `example.com`.

Auf Linux/Unix:

```
./subdomain—create mail -domain example.com
```

Auf Windows:

```
subdomain.exe—create mail -domain example.com
```

2. Der folgende Befehl erstellt die Subdomain `portfolio.example.com` und zur Verwaltung des Subdomain-Inhalts einen FTP-Account mit dem Login `portfolio` und dem Passwort `P4$$w0rd`.

Auf Linux/Unix:

```
./subdomain—create portfolio -domain example.com -ftp_user native -login
portfolio -passwd P4$$w0rd
```

Auf Windows:

```
subdomain.exe—create portfolio -domain example.com -ftp_user native -login
portfolio -passwd P4$$w0rd
```

Erstellen von Subdomains auf Unterordnern (nur Windows)

Befehl

Mit dem folgenden Befehl erstellen Sie eine Subdomain auf einem Unterordner:

subdomain <--create|-c> <*subdomain-prefix*> -domain <*parent-domain-name*> -hst_type virt [-www_root <*subdomain-root-directory*> -create_phys false]

wobei der Teil

- www_root <*subdomain-root-directory*> -create_phys false

dafür verantwortlich ist, eine Subdomain auf dem bereits existierenden Verzeichnis zu erstellen. Der Pfad wird mithilfe der `-www_root`-Option angegeben.

Beispiele

1. Der folgende Befehl erstellt die Subdomain `news.example.com` und ein physisches Verzeichnis `/httpdocs/news/` für den Inhalt der Subdomain.

```
subdomain.exe—create news -domain example.com -hst_type virt
```

2.Der folgende Befehl erstellt die Subdomain `blog.example.com` auf einem bereits existierenden Verzeichnis `/httpdocs/news/`. Nach der Erstellung ist der WordPress-Blog-Engine über die URL http://blog.example.com erreichbar.

```
subdomain.exe—create blog -domain example.com -hst_type virt -www_root
/httpdocs/wordpress -create_phys false
```

Verfügbare APS-Kataloge definieren

Um die benutzerdefinierte Konfigurationsdatei des APS-Katalogs zu importieren, verwenden Sie den Befehl im folgenden Format:

aps <--import-config|-ic> <*file-name*>

Die Konfigurationsdatei sollte sich in einem der folgenden Verzeichnisse befinden:

- Das temporäre Verzeichnis des Panels:
  - `/opt/psa/tmp` auf Debian, `/usr/local/psa/tmp` auf anderen Linux\Unix-Systemen
  - `%plesk_dir%\tmp` auf Windows
- Das temporäre Verzeichnis des Betriebssystems:
  - `/tmp` auf Linux/Unix
  - `C:\WINDOWS\Temp` auf Windows

Weitere Einzelheiten zu dem Format der Konfigurationsdatei finden Sie in dem Abschnitt **Konfigurationsdatei von APS-Katalogen**. (auf Seite 49).

Beispiel

Die folgenden Befehle importieren die Konfigurationsdatei von APS-Katalogen `catalogs.ini`:

Auf Linux/Unix:

```
./aps -ic /tmp/catalogs.ini
```

oder

```
./aps -ic /usr/local/psa/tmp/catalogs.ini
```

Auf Windows:

```
aps -ic "C:\Program Files\Parallels\Plesk\tmp\catalogs.ini"
```

oder

```
aps -ic "C:\WINDOWS\Temp\catalogs.ini"
```

Installieren von Anwendungen

Für die Installation von Anwendungen verwenden Sie das `aps`-Dienstprogramm.

In diesem Abschnitt:

Importieren eines Anwendungspakets 225
Installieren einer Anwendung 228

Importieren eines Anwendungspakets

In diesem Abschnitt:

Import von hochgeladenen Anwendungspaketen 226
Anwendungspakete vom APS-Katalog herunterladen 227
Status eines Download-Tasks abfragen 228

Import von hochgeladenen Anwendungspaketen

Um ein hochgeladenes Anwendungspaket zu importieren, verwenden Sie den Befehl im folgenden Format:

aps <--import-package|-ip> <*file-name*>

Beispiel

Der folgende Befehl importiert das Paket SugarCRM-5.0.0-16.app.zip in das Panel.

Auf Linux/Unix:

```
./aps—import-package /home/root/SugarCRM-5.0.0-16.app.zip
```

Auf Windows:

```
aps—import-package "C:\Documents and
Settings\Administrator\Desktop\SugarCRM-5.0.0-16.app.zip"
```

Anwendungspakete vom APS-Katalog herunterladen

Um ein heruntergeladenes Anwendungspaket zu importieren, verwenden Sie den Befehl im folgenden Format:

aps <--download|-d> -package-name <name> [-catalog-url <URL>] [-package-version <version-number> -package-release <release-number> -package-vendor <vendor-name> -package-packager <packager-name>]

wobei

- der Teil

  `[-catalog-url <URL>]`

  dafür verantwortlich ist, den Ziel-APS-Katalog zu definieren (die URL muss exakt mit der aus der Konfigurationsdatei übereinstimmen). Falls nicht angegeben, wird das Paket von dem APS-Katalog heruntergeladen, welches an erster Stelle in der Konfigurationsdatei (auf Seite 49) aufgelistet wird.
- der Teil

  `[-package-version <version-number> -package-release <release-number> -package-vendor <vendor-name> -package-packager <packager-name>]`

  dafür verantwortlich ist, das genaue Anwendungspaket anzugeben, die in der `-package-name`-Option definiert wurde. Falls dieser Teil nicht in dem Befehl angegeben wird und in dem Ziel-APS-Katalog mehrere Pakete mit demselben Namen existieren, dann wird das Paket mit der höchsten Version und Release heruntergeladen.

Beispiel

Mit dem folgenden Befehl wird das Paket mit der höchsten Version und Release der WordPress-Anwendung von dem APS-Katalog auf http://apscatalog.com heruntergeladen.

Auf Linux/Unix:

```
./aps -d -package-name WordPress -catalog-url http://apscatalog.com
```

Auf Windows:

```
aps -d -package-name WordPress -catalog-url http://apscatalog.com
```

Status eines Download-Tasks abfragen

Um einen Download-Status abzufragen, verwenden Sie den Befehl im folgenden Format:

aps <--get-download-status|-g> *<task id>*

Beispiel

Der folgende Befehl fragt den Status des Download-Tasks mit der ID 16 ab.

Auf Linux/Unix:

```
./aps -g 16
```

Auf Windows:

```
aps -g 16
```

Installieren einer Anwendung

Bei der Installation einer Anwendung ist es erforderlich, dass Sie angeben, welches Paket als Quelle für die Installation verwendet werden soll. Das Paket wird entweder über die zugewiesene ID ermittelt, wenn das Paket in das Panel importiert wird oder über die Paketinformationen “package name-version-release-vendor-packager” (Paketname-Version-Release-Anbieter-Verpacker).

Wählen Sie einen der folgenden Schritte, um eine Paket-ID anzugeben:

- verwenden Sie die ID, die dem Paket beim Import in das Panel zugewiesen wurde (auf Seite 226)
- verwenden Sie die ID, die dem Paket beim erfolgreichen Download vom APS-Katalog zugewiesen (auf Seite 228)
- verwenden Sie die ID, die in den Informationen der verfügbaren Pakete angegeben wird (auf Seite 233)

Um die Paketinformationen “package name-version-release-vendor-packager” (Paketname-Version-Release-Anbieter-Verpacker) anzugeben, fragen Sie die Informationen zu Paketen ab, die zur Installation zur Verfügung stehen (auf Seite 233).

In diesem Abschnitt:

Installieren einer Anwendung auf einer Domain oder Subdomain..... 229
Verfügbare Pakete abrufen 233

Installieren einer Anwendung auf einer Domain oder Subdomain

Um ein hochgeladenes Anwendungspaket auf einer Domain/Subdomain zu installieren, verwenden Sie den Befehl im folgenden Format:

aps <--install|-i> <<*filename*>|->

```
<
```

<-package-id <ID>>|<-package-name <name> -package-version <version-number> -package-release <release-number> -package-vendor <vendor-name> -package-packager <packager-name>>

```
>
<
```

<-domain <domain name>>|<-subdomain <subdomain name>>

```
>
```

[installation options]

wobei

- der Teil

  ```
  <filename>|-
  ```

  verantwortlich ist für die Angabe der Anwendungseinstellungen (im XML-Format (auf Seite 231)), sofern die Anwendung über welche verfügt. Die Einstellungsdefinition wird als Befehlsparameter angegeben, entweder als Dateiname oder direkt in dem Befehl, der von `STDIN` abgefragt wird. In dem zweiten Fall sollte "–" als Befehlsargument angegeben werden.

  Details zu dem Format der Einstellungsdefinition finden Sie in dem Unterabschnitt **Einstellungsdefinition für Anwendungen** (auf Seite 231).

- der Teil

  ```
  <-package-id <ID>>|<-package-name <name> -package-version <version-number> -package-release <release-number> -package-vendor <vendor-name> -package-packager <packager-name>>
  ```

  ist verantwortlich für die Angabe des Pakets, von dem aus die Anwendung installiert wird. Das Paket wird entweder über die ID oder über Name, Version, Release, Anbieter und Packager ("Verpacker") ermittelt. Diese Paketeigenschaften können über den `--get-packages-list`-Befehl (auf Seite 233) abgefragt werden.

- der Teil

  ```
  <-domain <domain name>>|<-subdomain <subdomain name>>
  ```

  verantwortlich ist für die Angabe der Ziel-Domain oder Subdomain, wo die Anwendung installiert wurde.

Verfügbare Optionen

Option	Parameter	Beschreibung
`-ssl`	`true|false`	Gibt an, ob eine Anwendung über das HTTPS-Protokoll zur Verfügung stehen wird oder nicht.

Option	Parameter	Beschreibung
-url-prefix	*<prefix>*	Gibt die zur Domain/Subdomain zugehörige URL an, über welche die installierte Anwendung im Internet verfügbar sein wird. Beispiel: Wenn Sie beispielsweise eine Blog-Engine Anwendung installieren und das URL-Präfix "blog" angeben, wird die Anwendung unter http://domain-name/blog verfügbar sein.
-db-name	*<name>*	Gibt den Namen der Datenbank an, die während der Installation für die Anwendung erstellt wird.
-db-user	*<login name>*	Gibt den Namen des Datenbankbenutzers an, der während der Installation für die Anwendung erstellt wird.
-passwd	*<password>*	Gibt den Namen des Datenbankpassworts an, die während der Installation für die Anwendung erstellt wird.

Beispiele

1.Der folgende Befehl installiert das Anwendungspaket mit der ID 13 auf der Domain example.com, die Anwendung wird über die URL example.com/blog erreichbar sein, die Einstellung werden in der Datei WordPress.xml definiert und bei der Installation wird die Datenbank WordPress erstellt.

Auf Linux/Unix:

```
# ./aps -i /home/apps/WordPress.xml -package-id 13 -domain example.com -ssl false -url-prefix blog -db-name WordPress -db-user BlogAdmin -passwd P4$$w0rd
```

Auf Windows:

```
>aps -i "E:\apps\WordPress.xml" -package-id 13 -domain example.com -ssl false -url-prefix blog -db-name WordPress -db-user BlogAdmin -passwd P4$$w0rd
```

2. Der folgende Befehl installiert dieselbe Anwendung nur mit den erforderlichen Optionen und die Anwendungseinstellungen werden von STDIN abgerufen.

Auf Linux/Unix:

```
# cat /home/apps/WordPress.xml | ./aps—install - -package-id 13 -domain example.com -db-name WordPress -db-user BlogAdmin -passwd P4$$w0rd
```

Auf Windows:

```
>aps—install - -package-id 13 -domain example.com -db-name WordPress -db-user BlogAdmin -passwd P4$$w0rd < C:\temp\WordPress.xml
```

3. Der folgende Befehl installiert eine Anwendung aus dem Paket mit der ID 14 auf der Subdomain api.example.com, die Anwendung erfordert keine Datenbank.

Auf Linux/Unix:

```
# ./aps -i /home/apps/empty.xml -package-id 14 -domain api.example.com -ssl true
```

Auf Windows:

```
>aps -i "E:\apps\empty.xml" -package-id 14 -domain api.example.com -ssl true
```

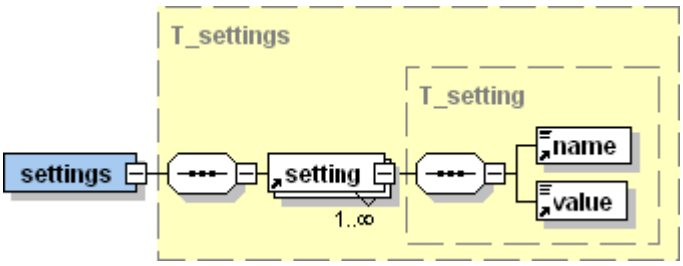
In diesem Abschnitt:

Definieren von Anwendungseinstellungen 231

Definieren von Anwendungseinstellungen

Falls eine Anwendung über Einstellungen verfügt, fragt das `aps`-Dienstprogramm nach festgelegten Anwendungseinstellungen aus einer Datei oder aus `STDIN`, wenn die Anwendungsinstanz auf einer Domain oder Subdomain installiert wird.

Die Anwendungseinstellungen sollten in einer XML-Datei mit der folgenden Struktur definiert werden:



- **settings**, *erforderlich*
  XML-Wurzelelement beinhaltet eine Reihe von Einstellungen
- **setting**, *erforderlich*
  Beinhaltet eine Definition von Anwendungseinstellungen
- **name**, *erforderlich*
  Gibt den Namen der Einstellung an, der von der Einstellungs-ID (`"setting[@id]"`) in der Metadaten-Datei des Anwendungspakets (`APP-META.xml` befindet sich in dem Anwendungspaket `*.app.zip`) abgeleitet wird.
- **value**, *erforderlich*
  Gibt den Wert für die Einstellung an.

Beispiel:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<settings>
  <setting>
    <name>admin_email</name>
    <value>johndoe@example.com</value>
    </setting>
    <setting>
        <name>admin_passwd</name>
        <value>p4$$w0rd</value>
    </setting>
    <setting>
        <name>site_name</name>
        <value>iScripts socialWare</value>
    </setting>
    <setting>
        <name>locale</name>
        <value>en-US</value>
  </setting>
</settings>
```

Verfügbare Pakete abrufen

Um die Informationen (ID, Name, Version, Release, Anbieter, Verpacker) zu den für die Installation verfügbaren Anwendungspakete abzurufen, führen Sie den folgenden Befehl aus:

aps <--get-packages-list|-gp>

Nimmt die folgenden Formen an.

Auf Linux/Unix:

```
./aps -gp
```

Auf Windows:

```
aps -gp
```